uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería Informática
2018-2019

Trabajo Fin de Grado

“Run away or shoot: diseño y desarrollo de un videojuego para PC en GameMaker”

Cristian López Reviriego

Tutor: Jorge Ruiz Magaña

Leganés, marzo de 2019

RESUMEN

En este Trabajo Fin de Grado se aborda el diseño y la implementación de un videojuego, denominado “Run away or shoot”, para plataforma de PC utilizando para ello la herramienta GameMaker: Studio.

El punto de partida del trabajo radica en la creciente y cada vez más relevante industria de los videojuegos, así como en la necesidad del ser humano de divertirse. En la fase de análisis se realiza un estudio de la competencia y quedan definidos los requisitos del juego. En la siguiente fase, de diseño, se detallan los diferentes elementos que componen el juego: objetos visibles, controles, mecánicas y música. En la fase de implementación, se explican a nivel técnico de qué forma se han programado los diferentes elementos definidos en el juego, entre los que destacan las técnicas de inteligencia artificial aplicadas en el comportamiento de algunos personajes del juego y los paradigmas de programación utilizados con la herramienta GameMaker, como son la programación dirigida por eventos y la programación orientada a objetos. Finalmente, se describe la fase de evaluación del juego, para la que se han empleado dos técnicas diferentes sobre un conjunto de usuarios de evaluación, haciendo un análisis cuantitativo y cualitativo de los resultados extraídos.

Finalmente, se realiza un estudio del coste económico y temporal que ha supuesto el desarrollo de este proyecto, así como de aquellos aspectos legales a tener en cuenta en caso de querer realizar una explotación económica del producto creado.

Palabras clave: Videojuego; Desarrollo software; GameMaker; Programación dirigida por eventos; Programación orientada a objetos; Inteligencia Artificial

AGRADECIMIENTOS

A mi tutor Jorge Ruiz Magaña, por los consejos y el tiempo que me ha dedicado durante todas las fases del trabajo, así como la motivación que hemos compartido por la temática elegida.

A mis padres, por darme su apoyo incondicional en estos años de carrera y la oportunidad de formarme en lo que me apasiona.

A mi hermano Óscar, que desde que éramos pequeños me transmitió su interés por la informática y que ha sido el mayor crítico de este trabajo, una labor imprescindible e impagable.

A mis amigos, a los que siempre han estado ahí y a los de “la comunidad”, por animarme a continuar hacia adelante y ayudarme a levantarme cada vez que me caigo.

A mis compañeros de universidad, en especial a Daniel y José Manuel, por aguantarme como compañero de prácticas y de los que aprendí tanto a nivel personal como profesional.

A todos los profesores que he tenido desde la educación secundaria, en especial a Máximo y Manuel.

A Soraya, por su comprensión y aportar luz en mi vida.

A Danny, por su colaboración y asesoramiento con los diferentes aspectos jurídicos tratados en este trabajo.

Por último, a todas las personas que me ayudaron con la evaluación del juego, aportando su honestidad y con un gran espíritu crítico.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. Motivación

El ser humano, ya desde sus primeras etapas de vida, tiene la necesidad de divertirse. Siempre se ha dicho que los juegos o, en general, las actividades que producen en el individuo algún tipo de sentimiento de diversión son especialmente útiles para la infancia, ya que promueven el desarrollo de las aptitudes físicas, la creatividad, la inteligencia emocional, las habilidades sociales, además de afianzar la personalidad y la introducción de valores de conducta. Los juegos son actividades muy útiles para conocer el entorno que nos rodea e introducir conceptos tan importantes como la empatía, la generosidad o la cooperación entre las personas.

Sin embargo, se tiende a pensar que a medida que las personas alcanzan la etapa de adulto, la diversión o el entretenimiento es algo innecesario y no tan importante para el equilibrio emocional y la salud del ser humano. Realmente esto no debería ser así, ya que precisamente el ocio saludable es una de las mejores actividades para liberar la mente del estrés cotidiano, de la rutina de las responsabilidades diarias y, en definitiva, mantener un equilibrio en el estilo de vida que muchas veces se impone en la sociedad actual.

Un gran ejemplo de filosofía de diversión, sin discriminar ningún tipo de edades, es la que aplica la compañía japonesa de videojuegos Nintendo y que se explica en el artículo escrito por Bella [1], en el que se afirma que su filosofía está en crear productos para generar sonrisas en los jugadores y para la que una de sus máximas personalidades, Shigeru Miyamoto, definió claramente con la frase “Creo que dentro de cada adulto reside el corazón de un niño. Lo único que nos hemos convencido gradualmente de que tenemos que actuar más como adultos”.

Como consecuencia de lo expuesto anteriormente y dejando a un lado aspectos más abstractos, es una realidad que la industria de los videojuegos está en auge. Si nos fijamos en el ámbito nacional, la industria de los videojuegos facturó en el año 2017, según informes redactados por la Asociación Española de Videojuegos (AEVI) [2], 1.359 millones de euros, viendo esta cifra incrementada respecto al año anterior en casi un 17%. Comparando estas cifras con las obtenidas en ese mismo año por otros sectores de la cultura y el entretenimiento, como son la industria del cine, que facturó 597 millones de euros, o la industria de la música, que facturó 232 millones de euros, podemos resaltar aún más la relevancia de este sector y que en muchas ocasiones no se aprecia en nuestra sociedad.

Esta posición de relevancia que tiene la industria de los videojuegos en la sociedad se debe, en mayor medida, al gran número de jugadores existente y a la notable variedad de perfiles de jugadores que consumen videojuegos de forma regular o casual. En el mismo informe al que se hace referencia anteriormente se ofrece un dato bastante elocuente respecto a este hecho, y es que se afirma que el 44% de los españoles cuya edad está comprendida entre los 6 y los 64 años juegan a videojuegos, aunque sea de forma casual. Teniendo en cuenta los datos de población publicados por el Instituto

Nacional de Estadística para el año 2017 [3], se concluiría que, aproximadamente, 15.5 millones de españoles juegan a videojuegos.

Asimismo, en el propio informe de AEVI, se dan datos sobre la distribución de la población de jugadores, cuantificándolos por género y franjas de edad:

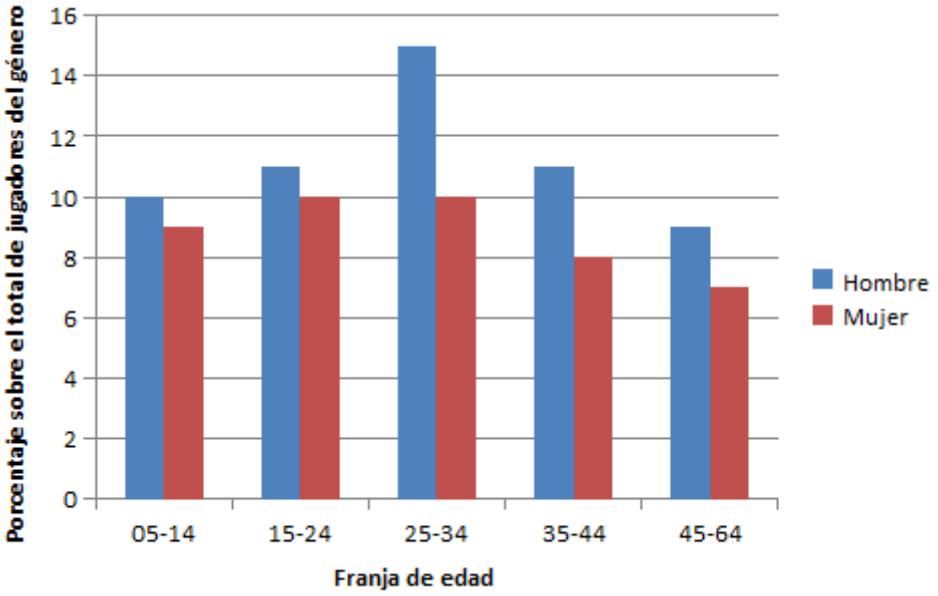


Fig. 1. Gráfico de porcentaje de jugadores por edades y género

Observando el gráfico anterior, tenemos que la mayor parte de jugadores respecto al total de jugadores e incluyendo ambos géneros, se encuentran en la franja de entre los 25 y 34 años de edad. Se puede concluir, además, que existe un prejuicio bastante claro respecto a la franja de edad predominante del perfil de un jugador de videojuegos. Habitualmente se dice que los jugadores de videojuegos mayoritariamente son adolescentes y jóvenes sin una responsabilidad laboral y familiar. En parte, el gráfico da la razón a esto, en la medida en que una gran parte de los jugadores corresponden a esas franjas de edad. Sin embargo, existe otro gran grupo de jugadores con edades superiores a los 25 años y en todas las franjas de edad analizadas. Gracias a esta información se puede concluir que existe una gran variedad de perfiles de jugadores, y potenciales clientes dentro de la industria de los videojuegos.

Es cierto que el mercado actual de videojuegos dispone de una oferta muy amplia para cubrir a todo tipo de perfiles de jugadores. Sin embargo, es importante destacar ese perfil de jugadores que disponen de menos tiempo para dedicar al entretenimiento, ya sea por motivos laborales y/o familiares, y que no por ello sean menos relevantes a la hora de proponer y diseñar nuevos juegos.

Un concepto muy interesante respecto a un tipo de jugador específico, y para los que se ha desarrollado una buena parte de la industria de los videojuegos, es el que se conoce como jugador casual, del que ya se hablaba en el año 2008 en la Conferencia de Desarrolladores de Juegos (GDC) y resumido por Sheffield [4]. En dicha conferencia, los organizadores definieron el concepto de juego casual como “aquel juego dirigido a personas para las que los videojuegos no son un área importante de interés, o al menos,

de mucho menor interés que para un jugador habitual". De igual forma se llegó a una serie de características principales que suelen compartir los juegos casuales, como son las siguientes:

- Bajas barreras de entrada: bajos tiempo de carga, ausencia de instalaciones complejas y tutoriales largos.

- Indulgentes: dar ciertas pistas y no penalizar en exceso los errores cometidos por el jugador durante el transcurso de una partida, al menos en los niveles más bajos de dificultad o en los primeros niveles de juego.

- Sesiones cortas de juego: una sesión de juego no debería requerir más de 10-15 minutos.

- Altamente rejugable: si el juego es de duración corta tendrá que ser rejugable para justificar su compra.

- Profundidad de juego revelada de forma gradual: si el juego va introduciendo nuevos elementos que sea de forma gradual y explicada para facilitar su entendimiento al jugador.

El problema que se pretende dar solución con este Trabajo Fin de Grado es el del diseño e implementación de un videojuego que ofrezca una opción de mercado para ese perfil de jugadores casual y dirigido, sobre todo, a jugadores dentro una franja de edad a partir de los 25 años en adelante, que no tengan una gran exigencia tanto por los gráficos del videojuego sino por el aspecto más jugable y de diversión.

1.2. Objetivos

Con la realización de este Trabajo Fin de Grado, en el que se pretende el diseño y desarrollo de un videojuego con las características comentadas en el subcapítulo anterior se busca lograr los siguientes objetivos:

1- Ampliar la oferta de juegos casual existentes en el mercado

El videojuego a desarrollar con este trabajo estará dirigido principalmente a un perfil de jugadores que siguen disfrutando de los videojuegos, pero que por motivos laborales o familiares no disponen de tanto tiempo para dedicarle a esta opción de ocio. Se ha detectado una oportunidad de negocio dentro del mercado de los videojuegos casuales que se pretende aprovechar.

2- Crear un juego de fácil portabilidad

Se buscará que, de forma sencilla, se pueda exportar el videojuego a otras plataformas que también tienen bastante interés para el perfil de jugadores más casuales, como podrían ser los teléfonos móviles o videoconsolas portátiles. Para lograr este objetivo tendrá mucha importancia la herramienta de programación elegida, que para el presente trabajo será Game Maker: Studio 2. Dicha herramienta, entre otras características que se comentarán en profundidad en el capítulo 2, permite desplegar el juego creado en diferentes plataformas en función de la licencia adquirida.

3- Crear un juego fácilmente escalable

Se buscará que el diseño del juego y su posterior implementación permitan un escalado sencillo, tanto en escenarios, tipos de enemigos, objetos, puzles y otros elementos del juego.

4- Aplicación de una metodología de desarrollo software tradicional

Es interesante ver de qué forma se pueden aplicar las fases típicas de análisis, diseño, implementación y evaluación, para un producto menos habitual en la formación académica de un ingeniero informático, y que pueda servir de referencia para proyectos similares futuros.

1.3. Marco regulador

Durante la realización de este Trabajo Fin de Grado, atendiendo a la naturaleza del producto creado, se deben tener en cuenta una serie de aspectos jurídicos. Dichos aspectos se desarrollarán en mayor profundidad en el capítulo 8 de este mismo documento. No obstante, en virtud de la perspectiva técnica del presente trabajo, dicho capítulo se abordará como una mera aproximación al plano jurídico.

En primer lugar, se hará un breve análisis de la situación actual en que se encuentra la legislación española y europea acerca de la propiedad intelectual y de cuál sería la mejor manera de conseguir una protección del videojuego desarrollado como el que se va a desarrollar con este trabajo.

En segundo lugar, dada la interacción necesaria con usuarios finales y usuarios de evaluación, es necesario tener en cuenta el cumplimiento de las obligaciones jurídicas que impone la legislación que regula el tratamiento y utilización de los datos de carácter personal.

En tercer lugar, se comentarán una serie de recomendaciones legales encaminadas a la posible explotación comercial del videojuego. En concreto, se propondrán las posibles vías y estructuras jurídicas para la posible explotación comercial del videojuego, ya sea a través de la explotación directa o la cesión a terceros de los diferentes elementos que configuran la idea del producto.

1.4. Entorno socio-económico

El estudio que se realiza en este trabajo de las consecuencias económicas y sociales esperadas del producto generado se centrará en dos subcapítulos, que por solapamiento con otras áreas temáticas del documento y para una mejor organización se ha optado por no separarlos en un capítulo en concreto.

En primer lugar, se realiza un estudio de la industria de los videojuegos dentro del ámbito nacional, dándole un mayor énfasis al crecimiento económico que ha tenido en los últimos años y la amplia variedad de perfiles de jugadores que existen en la sociedad que consumen este tipo de productos de ocio. Este estudio se realiza en el subcapítulo “2.1.1 Industria de los videojuegos”.

Respecto al producto generado, se detallará el presupuesto de la elaboración del presente Trabajo Fin de Grado en el subcapítulo “7.3 Presupuesto”.

1.5. Estructura del documento

A continuación, se enumeran y describen los diferentes capítulos de los que consta el presente documento.

En el capítulo “1. Introducción”, se comentan las diferentes razones, atendiendo al contexto social y económico, que motivan la realización de este Trabajo Fin de Grado. Una vez planteado el problema y la solución propuesta, se describen los objetivos que se pretenden lograr con la construcción de dicha solución y, en general, con la realización de este trabajo.

En el capítulo “2. Estado del arte”, se realiza un estudio de diferentes temáticas que tienen relación con el producto a desarrollar, como son la industria de los videojuegos y el uso de inteligencia artificial en los videojuegos. También se describen en profundidad algunas de las herramientas utilizadas durante la realización de este trabajo, entre las que se encuentran Game Maker: Studio 2, Balsamiq, BOUML y Git/Gitlab.

En el capítulo “3. Análisis de la solución”, se define el alcance del producto a desarrollar, especificando para ello una lista de requisitos que la solución deberá cumplir. Además, una parte de este capítulo se dedica a realizar un estudio de la competencia y productos sustitutivos que guarden alguna relación con la solución propuesta.

En el capítulo “4. Diseño de la solución”, se describen de forma más detallada todos los elementos que componen el juego, ya sean algunos aspectos abstractos, como la sinopsis, la mentalidad que se busca en el jugador y las mecánicas de juego, o elementos más perceptibles como los prototipos de pantallas y niveles de juego, la música o efectos de sonido necesarios. En este capítulo también quedará definido un diagrama de clases que intervienen en el juego diseñado, definiendo atributos, métodos y relaciones entre las diferentes clases.

En el capítulo “5. Implementación de la solución”, se detalla de qué forma se ha resuelto cada uno de los aspectos del juego teniendo en cuenta su programación, así como los principales problemas encontrados. Para ello se ha dividido este capítulo en varios apartados, entre los que se encuentran algunos como la creación de pantallas, la creación de niveles de juego, la creación de los personajes, la creación de los ítems o la programación de la inteligencia artificial de los enemigos.

En el capítulo “6. Evaluación de la solución”, se describe el proceso de evaluación del producto por parte de un grupo de usuarios, teniendo en cuenta diferentes perfiles de usuario, experimentos y técnicas de evaluación. Al final del capítulo, se extraen una serie de conclusiones a partir de los resultados de la evaluación del producto.

En el capítulo “7. Metodología y planificación del proyecto”, se describe la metodología de desarrollo utilizado para la realización del juego, detallando además los costes económicos y temporales de la totalidad de este Trabajo Fin de Grado.

En el capítulo “8. Marco regulador”, se mencionan aquellos aspectos legales que habría que tener en cuenta en caso de querer realizar una explotación económica del producto desarrollado en este trabajo. Además, se dedica un apartado a temas de propiedad intelectual, que se deberían tener muy en cuenta en caso de querer hacer un uso comercial del producto. Otros aspectos analizados son el tratamiento de datos que se realizan en la fase de evaluación del producto y algunos consejos legales acerca de la protección de la patente del proyecto.

En el capítulo “9. Conclusiones”, se trata de dar una justificación del cumplimiento de cada uno de los objetivos planteados al principio del trabajo.

En el capítulo “10. Trabajo futuro”, se plantean una serie de líneas de trabajo relacionadas con este proyecto, que podrían ser utilizadas por estudiantes para futuros trabajos académicos y de investigación.

En el capítulo “11. Lista de referencias”, se listan las fuentes o referencias bibliográficas que han sido utilizadas para la realización de este trabajo, y que se enlazan convenientemente en este documento.

2. ESTADO DEL ARTE

2.1. Estado del arte respecto del producto

En esta primera parte del estudio del estado del arte para este trabajo se analizan aquellos temas relacionados con el producto que se va a desarrollar, y que se consideran de una relevancia sustancial para entender el contexto actual con el que estamos tratando.

2.1.1. Industria de los videojuegos

La industria de los videojuegos se podría definir como el sector económico dedicado al desarrollo y la comercialización de videojuegos. De forma consecuente, queda definido un trabajador de la industria de los videojuegos como aquella persona que está involucrada en alguna de las fases de desarrollo de un videojuego, o en alguna de las fases de su posterior comercialización.

Algunos ejemplos de trabajadores de la industria de los videojuegos podrían ser: un guionista, que participa en una primera etapa de diseño del juego; un diseñador gráfico o un programador, que participan en la fase de implementación; un probador de videojuegos, que participa en etapas finales de desarrollo; o incluso el propio dependiente de una tienda de videojuegos, que vende el producto final al usuario. Es, por tanto, un hecho la gran cantidad de trabajadores que engloba esta industria y la importancia que tiene en la sociedad actual.

Dentro del ámbito nacional, si atendemos a los datos ofrecidos por la Asociación Española de Videojuegos (AEVI) en un informe económico presentado en enero de 2018 [5], podemos extraer una serie de conclusiones esclarecedoras de la dimensión de la industria de los videojuegos en nuestro país.

En primer lugar, si nos fijamos en el impacto económico de la industria de los videojuegos en el año 2016 para la economía española, teniendo en cuenta la producción efectiva, se correspondería con un 0,11% del PIB. Las cifras concretas de empleados directos de esta industria y que se muestran en el informe serían de 8.790 personas y un valor añadido de 503 millones de euros.

Un hecho muy revelador del impacto económico que tiene en el conjunto de los sectores económicos en España es que por cada euro que se invierte en la industria de los videojuegos se tiene un impacto de 3 euros en el total de los sectores nacionales y que por cada nuevo empleado en el sector de los videojuegos se crean 2,6 empleados en otros sectores. Estos datos implican la gran interrelación existente entre los distintos trabajos que se realizan en la industria de los videojuegos y otros sectores económicos, es decir, la interdisciplinariedad del sector de los videojuegos.

Si atendemos a los datos de facturación de videojuegos en España, tanto en formato físico como online en los años 2016 y 2017 podemos observar el crecimiento en la demanda de videojuegos que se está produciendo en estos momentos en el sector y que se muestran en la siguiente tabla:

	2016	2017
Facturación Física:	781 M€	885 M€
Facturación Online:	382 M€	474 M€
Total:	1.163 M€	1.359 M€

Tabla 1: Facturación en la industria de los videojuegos en España

2.1.2. Juegos *indie*

Los videojuegos independientes, o mayormente conocidos como juegos *indie*, no tienen una definición en la que todas las personas del sector de los videojuegos se hayan puesto totalmente de acuerdo. Como se indica por Grill [6], existen principalmente dos vías discrepantes acerca de qué aspecto sirve para discriminar qué videojuego se podría considerar independiente y cuál no.

Por una parte, tenemos el grupo de personas que piensan que el aspecto de financiación del desarrollo del videojuego es el punto determinante. En este caso, si la fuente de financiación económica del videojuego es el propio desarrollador entonces estaríamos antes un videojuego *indie*. Un juego no independiente quedaría definido, por tanto, como aquel que ha recibido financiación económica por parte de terceras empresas, como empresas publicadoras, o personas no implicadas de una forma clara en el desarrollo del mismo.

En el otro lado, tenemos la vertiente que se decanta por definir como videojuego independiente aquel que en el que las decisiones de diseño se han llevado enteramente por el equipo de desarrolladores, sin seguir al pie de la letra una serie de directrices por parte de personal de dirección de una compañía.

En general, y para el trabajo que nos ocupa, se tomarán ideas de ambas vías y quedará definido como videojuego *indie* aquel videojuego que ha sido diseñado sin seguir directrices determinadas por terceras personas ajenas al desarrollo del videojuego, y que además, ha sido enteramente financiado por el propio personal de desarrollo.

Es interesante analizar la evolución histórica que han seguido los videojuegos independientes para entender el porqué del auge que tienen en la actualidad, y que trata de explicar Cobbett [7]. La primera época dorada de este tipo de videojuegos se remonta a los inicios de la década de los años 90, con el auge de una nueva forma de distribución de software denominada *shareware*, donde las personas no tenían que abonar el precio de una licencia completa para poder probar un determinado software que estaba generalmente limitado en tiempo de uso o de funcionalidades. Algunos de los juegos *indie* más importantes de esos años, siguiendo ese modelo de distribución, y que además fueron revolucionarios para la industria del videojuego por los avances técnicos que aportaron fueron Wolfenstein 3D (1992) y Doom (1993).

Fig. 2. Ejemplos de pantallas de juego para Wolfenstein 3D (1992) y Doom (1993)

En los siguientes años, y debido principalmente al crecimiento en las prestaciones de las tarjetas gráficas tuvo como consecuencia que el desarrollo de videojuegos fuese una tarea mucho más compleja para desarrolladoras independientes. Sin una financiación económica potente vieron reducidas sus posibilidades de competir contra compañías ya asentadas en el sector.

Esta tendencia negativa de los juegos independientes tuvo su punto de inflexión con la aparición en la última década de herramientas que ayudan al desarrollo de videojuegos como Adobe Flash, Game Maker o Unity 3D. A este cambio de tendencia también influyeron otros factores como la popularización de internet, un medio a través del cual se distribuye y consume la mayor parte de los juegos independientes, y la aparición de nuevas plataformas de juego, como *smartphones* y *tablets*, que amplían las posibilidades de desarrollo y el público objetivo para las compañías desarrolladoras.

Actualmente tenemos algunos ejemplos de grandes éxitos de ventas dentro de los videojuegos independientes, como por ejemplo el juego de plataformas Cuphead (2017) desarrollado por la compañía StudioMDHR, que apenas una semana después de publicarse ya había vendido más de 300.000 copias del juego y en dos meses llegaba a la cifra de 2 millones de copias vendidas entre las diferentes plataformas de juego. Otro juego similar en cuanto al éxito que ha obtenido sería el simulador de granja Stardew Valley (2016) desarrollado enteramente por un único desarrollador, el estadounidense Eric Barone, que alcanzó los 3 millones de copias vendidas tras un año de su lanzamiento.

Fig. 3. Ejemplos de pantallas de juego para Cuphead (2017) y Stardew Valley (2016)

2.1.3. Inteligencia artificial en los videojuegos

El concepto de inteligencia artificial genera bastante controversia incluso entre las propias personas especializadas del sector. Existen multitud de debates acerca de qué se puede considerar como algo inteligente o como un simple algoritmo programado por una persona y ejecutado posteriormente por una máquina. Para la realización de este trabajo tomaremos como válida la definición dada por la RAE, que dice que inteligencia artificial es la disciplina científica que se ocupa de crear programas informáticos que ejecutan operaciones comparables a las que realiza la mente humana, como el aprendizaje o el razonamiento lógico.

Para completar la anterior definición, se debe explicar el concepto de test de Turing, prueba propuesta por Alan Turing en el año 1950 [8], que consiste en distinguir por parte de un humano quién es la máquina y quién es el humano mientras estos mantienen una conversación textual en lenguaje natural. Con esta prueba no se busca evaluar la capacidad de respuesta de la máquina, sino su similitud en comportamiento con el de un ser humano.

Dentro del ámbito de que nos ocupa en este trabajo, quedaría definido inteligencia artificial en los videojuegos como la aplicación de técnicas en videojuegos para simular comportamientos inteligentes en personajes que no estén controlados por el jugador, como podrían ser los enemigos, los jefes finales o personajes de apoyo para el jugador.

El uso que se ha hecho de las técnicas de inteligencia artificial en los videojuegos a lo largo de la historia es algo que se remonta a su propio nacimiento. Ya en la década de los años 50, el estadounidense Arthur Samuel comenzó a aplicar algoritmos de búsqueda en árbol para encontrar la solución del popular juego de tablero de las damas o Claude Shannon, que utilizó técnicas similares para el ajedrez, introduciendo nuevos conceptos como la función de evaluación a partir del valor de las piezas que tenía un jugador y de la movilidad de estas.

Durante las décadas de los años 60 y 70, a medida que se iban desarrollando juegos más complejos y con mecánicas novedosas hasta la fecha, los problemas que se debían resolver mediante las técnicas de inteligencia artificial fueron cambiando. Un ejemplo de ello es el problema que se planteaba en el juego Space Invaders (1978), donde el objetivo del jugador era eliminar oleadas de alienígenas utilizando un arma láser y conseguir el mayor número de puntos posibles. En este juego se aplicaron técnicas de aprendizaje automático durante el desarrollo de una partida en los niveles más complejos del juego, en la medida que la máquina estudiaba los patrones de movimiento del jugador para tratar de predecir su posición y dirigir los ataques hacia esas zonas del nivel.

Fig. 4. Ejemplo de pantalla de juego para Space Invaders (1978)

En la siguiente década, en los años 80, destaca la aparición del concepto de personalización de los personajes no controlados por el jugador. El objetivo de esta personalización era dotar de diferentes comportamientos o de diferentes grados de inteligencia a los personajes según su apariencia o su color, mejorando de esta forma la experiencia de juego del jugador. Igualmente, empezó a tratarse un nuevo tipo de problema aplicando técnicas de inteligencia artificial como era el de la búsqueda de caminos o *pathfinding*, característica clave del conocido juego del Pac-Man (1980). En este juego, el objetivo del jugador era comerse todos los puntos localizados a lo largo de un laberinto evitando ser capturado por unos fantasmas. El problema de *pathfinding* se daba en el punto de vista de los fantasmas, cuyo comportamiento y características diferían ligeramente unos de otros, pero que en definitiva debían localizar y perseguir al jugador a lo largo del laberinto.

Fig. 5. Ejemplo de pantalla de juego para Pac-Man (1980)

A partir de la década de los años 90, y hasta la actualidad, se podría considerar como el clímax de la aplicación de técnicas de inteligencia artificial en los videojuegos. Esto es debido, sobre todo, al gran crecimiento del sector tanto en plataformas como en la amplia variedad de juegos producidos a lo largo de estos últimos años. La consecuencia de esta multitud de géneros y juegos existentes es la posibilidad de aplicar técnicas de inteligencia artificial a problemáticas nuevas y en entornos diferentes a los habituales.

Algunos de los ejemplos más interesantes de aplicación de inteligencia artificial en estas últimas décadas podrían ser los comportamientos inteligentes de las criaturas con las que interactuaba el jugador en el juego Black & White (2001), un juego muy novedoso para la época donde se aplicaban técnicas de aprendizaje por refuerzo en la forma en la

que el propio jugador podía calificar una acción que realizaban estas criaturas como buenas o malas, ejecutando una acción de premio o de castigo.

Fig. 6. Ejemplo de pantalla de juego para Black & White (2001)

En otro de género de juegos muy populares en estos últimos años, como son los FPS o juegos de tiros en primera persona, también se han ido aplicando nuevas técnicas de inteligencia artificial en el comportamiento de los enemigos, como por ejemplo la aplicación del paradigma híbrido de agentes. Se dice que es un paradigma híbrido porque tiene tanto componentes deliberativos, en la parte del trazado de un plan de actuación en una situación de juego determinada, como componentes reactivos, en la parte en la que responde a las actuaciones del jugador como, por ejemplo, tratando de esquivar los tiros y poniéndose a salvo. Algunos ejemplos de juegos FPS pioneros en la aplicación de este tipo de técnicas son GoldenEye 007 (1997) y Half Life (1998).

Fig. 7. Ejemplo de pantallas de juego para GoldenEye 007 (1997) y Half Life (1998)

En la actualidad, nos encontramos ante un panorama de enormes posibilidades, tanto en la aplicación de nuevas técnicas de inteligencia artificial a problemas ya estudiados en juegos del pasado como de investigación de nuevos problemas sobre los que poder aplicar técnicas ya utilizadas y ver cómo responden. La gran variedad de técnicas de inteligencia artificial existentes, como algoritmos de búsqueda, agentes, redes neuronales, técnicas de *pathfinding*, generación aleatoria de niveles o mapas, algoritmos genéticos y evolutivos, permiten en muchos casos la aplicación simultánea de varias técnicas de forma combinada.

El objetivo último es conseguir comportamiento cada vez más complejos y adaptados al nivel del jugador que permitan mejorar la experiencia de juego y, en definitiva, su grado de diversión.

2.2. Estado del arte respecto de la técnica

En esta segunda parte del estudio del estado del arte para este trabajo se analizan aquellas herramientas o técnicas que se van a utilizar para el desarrollo del producto final, y que se consideran más relevantes para entender el trabajo en toda su completitud.

2.2.1. Git/GitLab

Antes de pasar a definir de forma concreta qué es Git y sus principales funcionalidades es importante entender perfectamente que es un sistema o software de control de versiones (CVS). Por sistema de control de versiones se entiende una aplicación informática que permite realizar un seguimiento de la historia de una colección de archivos y además incluye la funcionalidad de revertir el conjunto de archivos a una versión anterior.

Por versión de la colección de archivos se entiende como el estado de todos los archivos y ficheros para un instante de tiempo concreto. Generalmente, los archivos y ficheros son código fuente que forman un programa informático, aunque se puede extrapolar a cualquier otro campo. Es importante señalar que tanto el historial de cambios como el conjunto de archivos y ficheros se almacenan en lo que se conoce como un repositorio.

Un caso especial es el de los sistemas de control de versiones distribuidos, que se caracterizan por almacenar el repositorio en un servidor central al que los distintos usuarios pueden acceder para obtener una copia de su contenido mediante un proceso de clonación [9] [10].

Git es un ejemplo de sistema de control de versiones distribuido, diseñado por Linus Torvalds, que permite a los diferentes colaboradores de un proyecto trabajar de forma asíncrona. Se trata de la herramienta de este tipo más utilizada por empresas y equipos de desarrollo.

Fig. 8. Logo de Git

Cuando se trabaja con Git se disponen de tres zonas de trabajo bien diferenciadas, que se describen a continuación:

1. Repositorio remoto: conjunto de cambios, archivos y ficheros almacenados en el servidor de Git. Sobre este repositorio no realizaremos modificaciones de forma directa al trabajar con los ficheros en nuestro equipo.

2. Repositorio local: conjunto de cambios, archivos y ficheros almacenados de forma local en nuestro dispositivo. Sobre este repositorio realizaremos modificaciones de forma directa al trabajar con los ficheros en nuestro equipo.
3. Área de preparación: zona intermedia donde se preparan las confirmaciones de los cambios realizadas sobre los ficheros y archivos antes de almacenarlas en el repositorio remoto.

El flujo de trabajo habitual cuando se utiliza Git queda reflejado en la siguiente figura, en la que se muestran algunos de los comandos más habituales cuando se trabaja con esta herramienta:

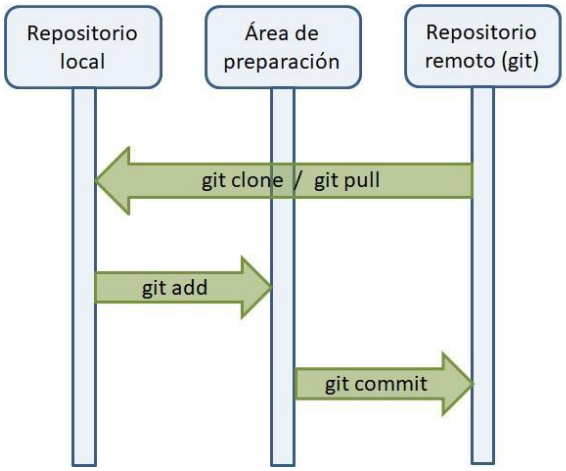


Fig. 9. Flujo de trabajo habitual en Git

Las principales funcionalidades y sus comandos asociados cuando se trabaja con Git son las siguientes:

- Creación de un repositorio: para esta finalidad se utiliza el comando *git init.* Una vez se crea el repositorio Git se genera una carpeta oculta de forma automática denominada ".git", que almacena una serie de ficheros con información interna del repositorio Git creado.

- Clonación de un repositorio: el objetivo de clonar un repositorio es obtener una copia del mismo en el directorio de trabajo local. El comando *git clone* seguido de la URL dónde esté almacenado el repositorio es el requerido para realizar esta función.

- Creación y confirmación de cambios: a medida que se van realizando cambios en los ficheros del directorio de trabajo local se pueden ir agregando al área intermedia o de preparación. Para ello se utiliza el comando *git add*. El siguiente paso es confirmar estos cambios del área de preparación al repositorio remoto de Git, para lo que se utiliza el comando *git commit*.

- Mostrar el estado del repositorio local y del área de preparación: mediante el comando *git status* se pueden visualizar los cambios realizados en los ficheros ubicados en el área de preparación respecto al repositorio local.

- Crear y administrar ramas: una de las opciones que permite Git es la creación de ramas, que queda definida como un contexto de trabajo determinado de un

directorio de trabajo. De un mismo repositorio se pueden tratar varias ramas, gestionando los cambios de forma separada, fusionarlas y eliminarlas. Mediante el comando *git branch*, con las distintas opciones de argumentos que se ofrecen podemos realizar todas estas funciones.

- Actualización del repositorio local a partir del repositorio remoto: mediante la ejecución del comando *git pull* se realiza una sincronización y fusión del directorio local a partir del remoto.

- Actualización del repositorio remoto a partir del repositorio local: mediante la ejecución del comando *git push* se permiten compartir los cambios confirmados o *commits* realizados en el repositorio local con el resto de usuarios que tienen acceso al repositorio remoto.

Una vez entendido qué es Git y sus principales funcionalidades se pasa a analizar la herramienta GitLab, que se va a utilizar durante la elaboración de este trabajo de forma conjunta a Git. GitLab es una herramienta de código abierto para el desarrollo de software colaborativo basado en Git. Las principales funciones que ofrece son: gestión de repositorios Git, realizar revisiones de código, seguimiento de tareas y creación de wikis para un proyecto.

La principal diferencia respeto a herramientas similares predecesoras, como GitHub, es la amplitud en los servicios que ofrece, sobre todo en temas relaciones con el desarrollo y operación del software (DevOps).

GitLab se ofrece como software libre y puede desplegarse en cualquier servidor. Una alternativa a esta opción es utilizar la herramienta a través de la página web *GitLab.com*. Esta vía es la más recomendable si se quiere ahorrar tiempo en configuración e instalación en un servidor [11] [12].

Fig. 10. Logo de GitLab

GitLab ofrece servicios para cada una de las fases del ciclo DevOps, cuyo objetivo es unificar las etapas de desarrollo y operación del software.

A continuación, se pasa a explicar las distintas funcionalidades que ofrece GitLab, en su versión gratuita, de acuerdo a cómo se ordenan en el menú una vez tenemos un proyecto creado en la herramienta:

- Proyecto: respecto a un proyecto creado, GitLab permite ver los archivos que contiene, versiones del mismo e información resumida del repositorio. Dentro de este mismo apartado se pueden consultar las acciones realizadas sobre el proyecto a modo de historial y ordenar de acuerdo a varios criterios. Otra opción

interesante son las métricas de tiempo empleadas en las distintas fases del ciclo de vida del proyecto, que se pueden consultar en la sección denominada *cycle analytics*.

- Tareas: se trata de uno de los principales servicios que ofrece GitLab. Se puede realizar un seguimiento personalizado y separado para cada una de las tareas en las que se puede descomponer el desarrollo del proyecto software. Para cada tarea se pueden abrir discusiones entre las personas asignadas, controlar los flujos de trabajo, establecer el tiempo estimado para la tarea, entre otras opciones. Para gestionar y visualizar de mejor forma todas las tareas que tiene un usuario asignadas se pueden consultar mediante unos tableros, en los que se organizan las tareas de acuerdo a unas etiquetas personalizables. Otro elemento que se puede definir trabajando con GitLab son los hitos, que sirven para controlar el progreso del desarrollo del software.

- Solicitudes de fusión de código: existe una sección separada en GitLab para gestionar las solicitudes de fusión de código de diferentes ramas de un mismo proyecto. Una de las opciones más interesantes respecto a esto es la posibilidad de discutir los cambios en el código entre las personas implicadas.

- CI/CD: otra de las funcionalidades que se ofrecen están destinadas a los procesos de integración continua y desarrollo continuo, útiles para automatización de procesos y creación de flujos de trabajo.

- Registro: GitLab permite gestionar las diferentes versiones del proyecto, pudiéndose realizar operaciones de descarga o eliminación de forma intuitiva.

- Snippets: existe una sección dedicada al análisis y revisión de fragmentos de código fuente y texto con otros miembros del proyecto, fomentando de esta forma la cooperación y el trabajo en equipo.

- Miembros: se ofrece una lista ordenable, de acuerdo a diferentes criterios, de las personas que forman parte del proyecto. De cada miembro se informa del nombre, rol que ocupa dentro del proyecto y la fecha de unión al mismo.

2.2.2. Balsamiq

Debido al auge de las aplicaciones web y el desarrollo de software para *smartphones* en los últimos años, las herramientas de soporte al diseño de prototipos de pantallas adquieren una mayor relevancia. En concreto, son una herramienta de vital importancia para las personas especializadas en HCI, que es la disciplina que estudia las interacciones entre personas y computadoras. Este rol de trabajadores, que se suelen denominar como diseñadores de interfaces o diseñadores de interacción, cada vez tiene un papel más importante en los equipos de desarrollo de software, ya que un diseño de pantallas eficaz e intuitivo para los usuarios puede ser el punto diferencial entre una aplicación exitosa y otra que no lo sea para el usuario final.

Balsamiq es un ejemplo de herramienta para el diseño de prototipos de pantallas enfocados en el tipo de aplicaciones descritas anteriormente [13]. En el presente trabajo esta herramienta será utilizada para la construcción de los prototipos de las pantallas del juego, que se describirán más adelante en el capítulo 4, dedicado al diseño del juego.

Fig. 11. Logo de Balsamiq

Actualmente, aquellas personas que quieran trabajar con Balsamiq disponen de una licencia de prueba de 30 días de duración en la que se ofrece la totalidad de funciones que ofrece la herramienta. Estas funciones van desde la creación de prototipos de pantallas de aplicaciones móviles, con los elementos de interfaz típicos de estas plataformas, a la creación de prototipos de aplicaciones web y de escritorio. Se trata de una herramienta bastante intuitiva de manejar para alguien familiarizado con las herramientas *drag & drop*. Los distintos elementos que componen el prototipo que se esté diseñando en esos momentos con una simple operación de arrastrar y soltar.

A continuación se muestran algunos ejemplos de prototipos que han sido realizados utilizando la herramienta Balsamiq, en un primer caso para una aplicación móvil y en el segundo caso para una página web:

Fig. 12. Ejemplos de prototipos de pantallas creados con Balsamiq

2.2.3. BOUML

Una herramienta CASE (*Computer Aided Software Engineering*) es un tipo de software utilizado para el soporte de desarrollo de software. Se pueden utilizar en cualquier fase del ciclo de desarrollo de un producto, como, por ejemplo, con la realización de diagramas de casos de uso en la fase de análisis o diagramas de flujo y diagramas de clases en la fase de diseño. Este tipo de herramientas son de gran utilidad para la labor

de un ingeniero de software ya que permiten, en muchos casos, traducir a código fuente una representación gráfica de, por ejemplo, un diagrama de clases.

BOUML es un ejemplo de herramienta CASE. Esta aplicación, que además de caracterizarse por ser gratuita desde su versión 7.0, permite la construcción de diagramas en formato UML 2 (*Unified Modelling Language*), que es el lenguaje de modelado de sistemas más utilizado en la actualidad por empresas de desarrollo de software. Este lenguaje define una serie de estándares sobre la construcción de los diferentes productos o diagramas generados durante el desarrollo de un software, entre los que se encuentran los diagramas mostrados en la siguiente figura [14]:

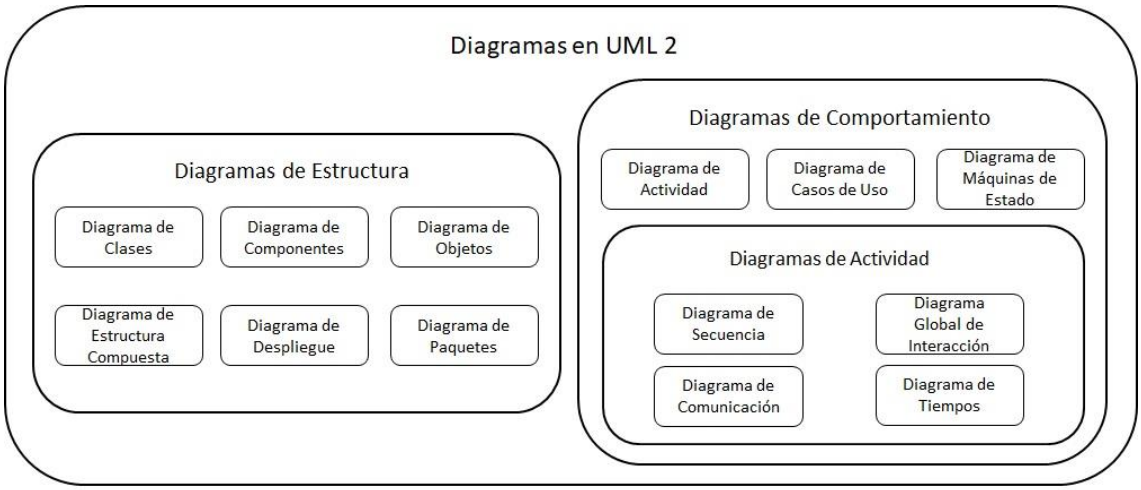


Fig. 13. Diagramas en UML 2

Una vez se realizan diagramas UML con esta herramienta, una opción que se dispone es la de generar el código resultante que traduce dicho diagrama a un lenguaje de programación concreto. BOUML permite generar código en lenguajes bastantes comunes en la actualidad como son C++, Java, Php, Python y MySQL [15].

La utilización de la herramienta es bastante intuitiva para el usuario, ya que la gran mayoría de acciones que se pueden realizar en un diagrama son con operaciones del tipo arrastrar y soltar. La gran mayoría de elementos que forman parte del diagrama que se construyendo en ese momento tienen una serie de propiedades que se pueden editar de forma bastante sencilla para los usuarios, como pueden ser, el sentido de las interrelaciones entre clases, los atributos y métodos de una clase o los argumentos de un determinado método de clase.

A continuación, se presenta un ejemplo de diagrama de clases generado con la herramienta BOUML, donde se puede observar también la apariencia que tiene la aplicación:

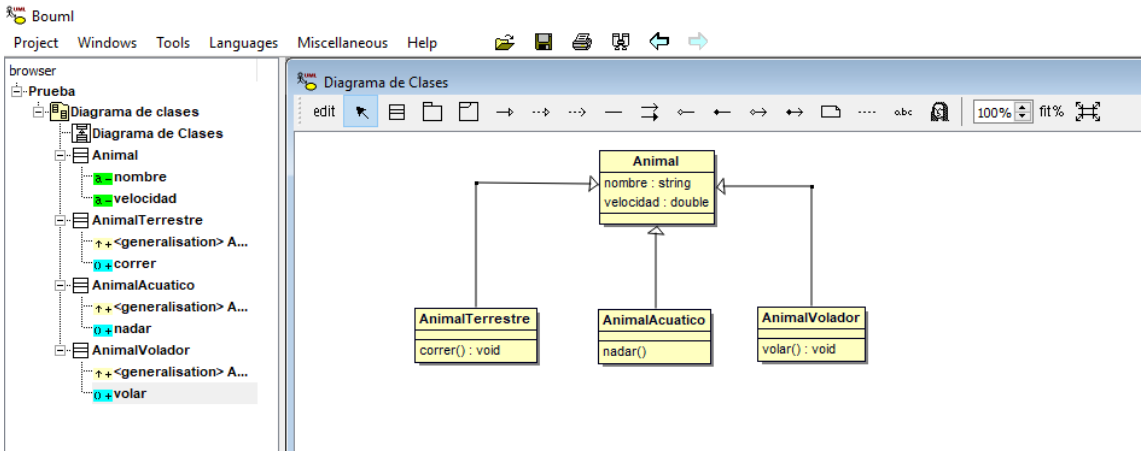


Fig. 14. Ejemplo de construcción de un diagrama de clases en BOUML

2.2.4. Game Maker Studio 2

Game Maker: Studio 2 es un entorno de desarrollo integrado (IDE) basado en un lenguaje de programación interpretado para el desarrollo de videojuegos dirigido a usuarios sin necesidad de tener unos amplios conocimientos en programación. Su primera versión fue liberada en 1999 y creada por Mark Overmars, profesor de programación de videojuegos para la universidad de Utrecht en los Países Bajos.

Su desarrollo corre a cargo de la compañía YoYo Games Ltd., con sede en Dundee (Escocia), aunque desde 2015 es una empresa subsidiaria de Playtech. El objetivo de la compañía es apoyar el desarrollo del producto Game Maker y formar una comunidad de desarrolladores y jugadores ocasionales que puedan ofrecer sus propios videojuegos en la web [16].

Actualmente se dispone de una versión totalmente gratuita con la que poder desarrollar pequeños videojuegos totalmente funcionales, aunque existen otras versiones de pago que amplían algunas de las características. Por ejemplo, adquiriendo una de estas versiones de pago se permite desplegar el videojuego en otras plataformas además del PC como son los teléfonos móviles, Nintendo Switch o PlayStation 4. Otras de las características más interesantes al adquirir una versión mejorada es la de disponer de recursos ilimitados a la hora de construir los videojuegos con la herramienta.

Fig. 15. Logo de Game Maker: Studio 2

A la hora de comenzar a diseñar un videojuego en Game Maker Studio es importante tener en cuenta que los paradigmas de programación que se utilizan cuando se trabaja con esta herramienta son el de programación orientada a objetos (POO) y la programación dirigida por eventos.

La aportación de la programación orientada a objetos en el desarrollo de videojuegos en Game Maker viene dada por el hecho de que todos los elementos que diseñamos y forman parte del videojuego se consideran objetos, cada uno con sus atributos y sus métodos. Estos objetos pueden ser los personajes, las casillas, los efectos visuales, los efectos sonoros, entre otros elementos del juego.

Respecto a la programación dirigida por eventos, afecta en la forma de diseñar y concebir el videojuego en que las diferentes partes de código de los diferentes objetos se van ejecutando a medida que se activan ciertos eventos. Un evento en Game Maker es un momento discreto durante el desarrollo de una partida del juego que es capturado y a los que se asocia una serie de acciones o sentencias de código. Por tanto, los eventos son la forma en la que el jugador es capaz de interaccionar con el desarrollo de la partida y, a su vez, los distintos elementos que componen la partida pueden relacionarse entre sí.

En la siguiente figura se presenta una lista de eventos disponibles en Game Maker y que se pueden asociar a cualquier objeto creado:

Fig. 16. Eventos disponibles en Game Maker

Un proyecto en Gam eMaker se compone de una serie de recursos que se distribuyen por categorías en el lado derecho del interfaz gráfico de la herramienta. Existe una gran variedad de tipos de recursos con los que se puede trabajar en el diseño del videojuego y que podemos ver en la siguiente ilustración:

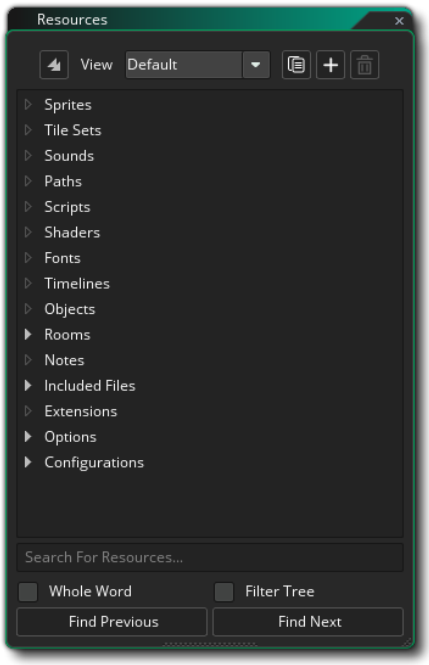


Fig. 17. Recursos disponibles en Game Maker

A continuación, se describen los recursos disponibles más relevantes y qué tipo de elementos representan a la hora del diseño de un videojuego en Game Maker [17]:

- Sprites: se trata de cualquier tipo de imagen que se va a utilizar en el proyecto. Puede ser desde una imagen fija o una animación, y se asocian a personajes, fondos, efectos, que se van a utilizar en la construcción del videojuego.

- Conjuntos de casillas: se trata de las diferentes casillas que se van utilizar, si procede, para cubrir el fondo de una determinada sala o pantalla del videojuego.

- Sonidos: se trata de cualquier efecto sonoro que se vaya a reproducir en el transcurso de una partida. Por ejemplo, el sonido producido al ser derrotados, después de conseguir avanzar de nivel, o tras resolver con éxito un determinado puzle.

- Rutas: se trata de un elemento para definir trayectorias o caminos a seguir por determinados objetos creados en el juego. Por ejemplo, la trayectoria parabólica a seguir por un objeto arrojado por el personaje principal o el movimiento de ciertos personajes no controlados por el jugador.

- Scripts: se trata de cualquier fragmento de código, en lenguaje GML, que se quiera reutilizar en diferentes partes del proyecto. Por ejemplo, puede ser útil crear un script para controlar efectos físicos como la fuerza de la gravedad sobre un salto de un personaje.

- Shaders: se trata de un recurso para realizar modificaciones sobre los gráficos de los distintos elementos que aparecen en el juego, a nivel de tarjeta gráfica, liberando de esta forma de carga computacional al procesador.

- Fuentes: se trata de un recurso utilizado para controlar el formato de letra que se va a utilizar en elementos de juego como pueden ser rótulos, texto de menús, mensajes informativos durante la partida.

- Líneas temporales: se trata de un recurso para controlar y agrupar secuencias de acciones que transcurren a lo largo del tiempo, asociadas a objetos concretos.

- Objetos: se trata de un elemento del juego en sí mismo, que controlan la interactividad y la lógica del juego. Pueden estar vinculados a un sprite (personaje, enemigos, ítems) o no estar vinculados a ningún sprite. Estos pueden ser, por ejemplo, elementos invisibles para controlar la lógica del juego, como un generador de enemigos aleatorio.

- Salas: se trata de un recurso del juego que cubre lo que sería una pantalla o nivel de juego. Una sala, además, se trata como un contenedor donde se instancian objetos de los disponibles en el árbol de recursos. Es importante señalar, que todo juego en Game Maker, debe disponer de al menos una sala y estas salas se pueden transitar mediante acciones.

En Game Maker se tiene como idea que el desarrollador del videojuego no tenga porqué ser un experto en programación. Para ello, además de facilitar muchas de las tareas de diseño y construcción del videojuego con los servicios que ofrece la propia herramienta, se ofrecen dos alternativas a la hora de programar las acciones asociadas a los diferentes eventos.

En primer lugar, se dispone de la opción de programación basada en un sistema de *Drag and Drop* (DnD). La idea de este tipo de programación está en seleccionar y construir la secuencia de acciones que se ejecutan cuando se activa un evento a partir de un conjunto predefinido. Dentro de este conjunto de acciones que se ofrecen se tienen algunas como instanciación de objetos, creación o modificación de variables, realizar movimientos, reproducción de sonidos, eliminar objetos, navegación entre salas, sentencias condicionales e iterativas, etc.

En la siguiente imagen se puede observar de qué forma se presenta al desarrollador esta técnica de programación en Game Maker:

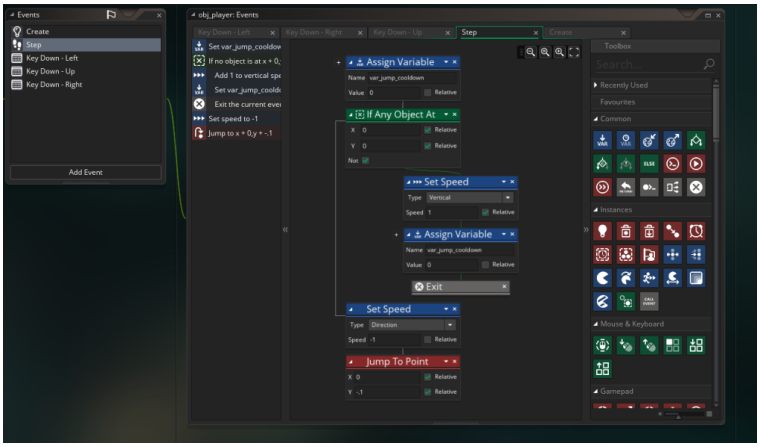


Fig. 18. Programación Drag and Drop en Game Maker

En segundo lugar, Game Maker ofrece un lenguaje de programación propio denominado GML (*Game Maker Language*) con el que poder codificar las acciones asociadas a los diferentes eventos producidos durante el desarrollo del juego. Se trata de un lenguaje muy flexible, basado en el lenguaje de programación C++, que se caracteriza por su legibilidad y simplicidad. Utilizando GML no es necesario definir las variables, sino que basta con asignarles un valor para comenzar a utilizarlas. Una característica muy interesante que ofrece Game Maker es disponer de una librería de funciones bastante amplia con la que facilitar la tarea del programador, como por ejemplo funciones para verificar si una tecla está siendo pulsada, instanciar y eliminar objetos, navegación entre salas o reproducción de sonidos.

A continuación, se presenta un ejemplo de codificación en GML:

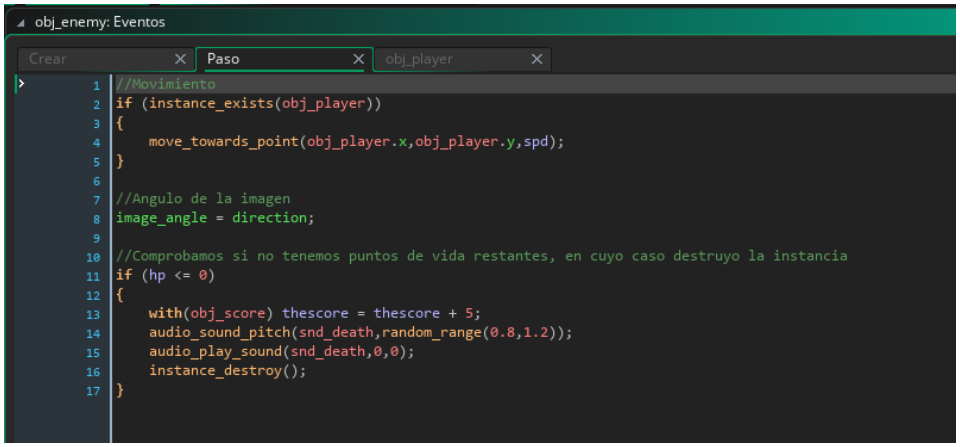


Fig. 19. Programación GML en Game Maker

3. ANÁLISIS DE LA SOLUCIÓN

3.1. Definición de la solución

Después de realizar un estudio del problema y observar que existe un gran potencial de jugadores casuales dentro del panorama nacional se propone como solución el diseño e implementación de un videojuego destinado a ese público objetivo. Dentro de las características analizadas que debería cumplir un juego casual, el foco estará puesto, sobre todo, en crear un juego sencillo e intuitivo, a la vez que sea divertido, y con sesiones de juego cortas de no más de 12 minutos de duración máxima por partida.

Con estas premisas se pasa a definir brevemente las características y mecánicas principales del juego propuesto como solución, que se titulará "Run away or shoot!":

1. Se trata de un juego que combina los géneros de acción y puzles.

2. El objetivo del jugador es conseguir huir de un complejo, que se encuentra completamente cerrado y con un cierto número de enemigos. Debe realizarlo dentro de un tiempo límite antes de que todo el complejo sea destruido. Para poder huir del complejo tendrá que encontrar y resolver ciertos puzles, a la vez que evitar ser derrotado por los enemigos que se encontrará por el complejo. Será decisión del jugador si opta por tratar de escabullirse entre los enemigos o intentar ganar tiempo enfrentándose a ellos.

3. En el caso de que el jugador necesite o desee enfrentarse a los enemigos, se dispone de una serie de objetos (armas, munición y otro tipo de objetos de utilidad) dispersos por el complejo y que le ayudarán en la resolución de la partida.

4. Los puzles a resolver durante el transcurso del juego serán planteados de una manera sencilla e intuitiva, buscando que su resolución al jugador no le lleve más de 2 minutos por puzle en término medio. Consistirán en descifrar un código numérico de cuatro cifras en el que cada cifra puede ir de valores desde el "1" hasta el "6" disponiendo para ello de intentos ilimitados. Tras confirmar el jugador un intento de resolución, se ofrecerán pistas acerca de cuántas cifras están bien puestas, y si alguna cifra forma parte del código correcto pero no está en el lugar adecuado. Estas pistas se darán de forma visual mediante un código de símbolos bastante intuitivo. Cuando el código numérico esté correctamente introducido el puzle se dará por resuelto y se alertará de ello al jugador.

5. Los enemigos tendrán un comportamiento inteligente en base a la información obtenida gracias al sentido de la vista y del oído. Pasarán de un estado normal de movimiento a un estado de hostilidad si tienen al alcance visual al jugador o si oyen ruidos, generalmente producidos por armas de fuego, a una distancia determinada. La programación de la inteligencia artificial se explicará más en detalle en el capítulo 5 dedicado a la fase de implementación del juego.

6. El juego buscará la diversión del jugador al combinar la sensación de tensión que generará sobrevivir dentro del complejo repleto de enemigos con un

equipamiento bastante limitado, junto con la resolución de los puzles usando habilidad de lógica. Todo en ello en tiempo real y teniendo siempre presente que deberá resolverse antes de que el contador de tiempo llegue a cero.

7. El juego tendrá varios modos de juego escalando la dificultad. Aunque la mecánica principal del juego no cambie, si cambiará el número de enemigos que aparecerán por el escenario, el número de puzles a resolver, y el propio escenario. En cualquier caso, el tiempo máximo de resolución de partida no cambiará.

8. Siguiendo una de las tendencias actuales en videojuegos, como es el resurgimiento de los juegos con gráficos de corte clásico o retro, también conocidos como juegos *Pixel Art*, se plantea el juego en esta línea. En este tipo de juegos la búsqueda del realismo en el apartado visual queda en un segundo plano, dándole una mayor prioridad a la jugabilidad. Este tipo de juegos utilizan modelos de gráficos pixelados o similares a juegos clásicos como Super Mario Bros (1985) o The Legend Of Zelda: A Link to the Past (1992), de los que se muestra a continuación un ejemplo a qué tipo de gráficos nos referimos:

Fig. 20. Gráficos en Super Mario Bros 1 (1985) y The Legend Of Zelda: A Link to the Past (1992)

9. Funcionará en plataforma PC, con sistema operativo Windows.

Una parte importante de la definición de la solución es tener en cuenta que se utilizarán los distintos servicios ofrecidos por la herramienta Game Maker: Studio 2.

3.2. Estudio de competencia y sustitutivos

Una vez determinada la solución al problema, en este apartado se pasa a realizar un estudio de aquellos productos competencia y/o sustitutivos existentes en el mercado actual de los videojuegos y de características similares teniendo en cuenta sobre todo género, mecánicas de juego, jugabilidad y apartado visual.

El estudio de productos competencia y sustitutivos se realiza de forma individual. Cada juego se analiza ofreciendo información general, como son el título, año de lanzamiento, género, desarrollador y una breve descripción acerca de en qué consiste. Además, se enumeran las mecánicas que más los caracterizan, destacando también los puntos más positivos y negativos de acuerdo a las principales fuentes de críticos especializados del sector. Finalmente, para tener una visión más clara de cómo es el

aspecto visual de cada producto se presenta una ilustración de una pantalla normal de juego.

3.2.1. Lone Survivor: The Director's Cut

- Mes de lanzamiento: Abril de 2012
- Género: Acción, Aventuras, *Indie*
- Desarrollador: Jasper Byrne
- Descripción: en este juego de aventura de terror psicológico controlamos a un hombre que es un superviviente aislado de una infección que ha convertido a la población mundial en agresivos mutantes. Durante el transcurso de la partida el jugador deberá ayudar al personaje protagonista a explorar el mundo que le rodea consiguiendo sobrevivir al mismo tiempo, encontrando comida y otro tipo de objetos que le ayudarán en su objetivo [18] [19].
- Mecánicas: supervivencia, exploración y navegación lateral en escenarios 2D, gestión de objetos, resolución de puzles, disparos.
- Puntos positivos: los entornos y la atmósfera de terror está muy bien conseguida, gracias tanto a la historia, al apartado visual y a los efectos de sonido. Realmente es un juego de supervivencia.
- Puntos negativos: la navegación por los distintos escenarios es un poco tediosa para el jugador; la mecánica de disparos no está del todo bien lograda.
- Aspecto visual:

Fig. 21. Ejemplo de pantalla de juego para Lone Survivor: The Director's Cut (2012)

3.2.2. Mark of the Ninja

- Mes de lanzamiento: Octubre de 2012
- Género: Acción, Aventuras, Indie
- Desarrollador: Klei Entertainment

- Descripción: en este juego de plataformas orientado al sigilo controlamos a un ninja que deberá avanzar por una serie de niveles, en los que existen áreas de luz y oscuridad. Será tarea del jugador esquivar y/o derrotar a una serie de enemigos aprovechando para ello la capacidad de ocultación en diferentes elementos del escenario y una serie de ataques silenciosos que puede realizar el personaje. Al completar cada nivel se ofrece al personaje la posibilidad de mejorar con la compra de objetos, la adquisición de nuevas habilidades y técnicas de combate [20] [21].

- Mecánicas: sigilo, plataformas con navegación lateral en 2D, ocultación utilizando elementos del escenario, gestión de objetos, adquisición de habilidades entre escenarios.

- Puntos positivos: la mecánica de sigilo está bien lograda, recompensando ese estilo de juego al jugador. Estéticamente está muy bien cuidado, con alto grado de rejugabilidad. Destaca también el establecimiento de un buen número de puntos de guardado para no frustrar al jugador en sus avances.

- Puntos negativos: los controles, en su versión para PC y con teclado, es bastante mejorable.

- Aspecto visual:

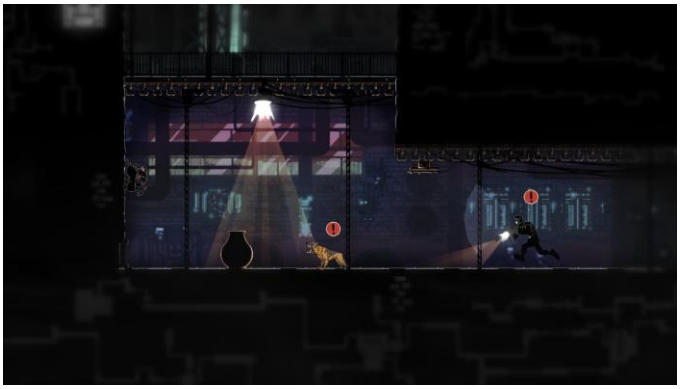


Fig. 22. Ejemplo de pantalla de juego para Mark of the Ninja (2012)

3.2.3. Stealth Inc: A Clone in the Dark

- Mes de lanzamiento: Julio de 2013

- Género: Acción, Indie

- Desarrollador: Curve Studios

- Descripción: en este juego de plataformas en 2D controlamos a un personaje que, haciendo uso de sus habilidades de sigilo, deberá escapar de una serie de niveles en los que se encuentran desplegados un conjunto de centinelas, cámaras de seguridad y otros elementos de alarma. El castigo para el jugador que sea detectado por algún sistema de seguridad del nivel en cuestión será la derrota inmediata [22] [23].

- Mecánicas: sigilo, plataformas con navegación lateral en 2D, planificación de secuencia de acciones, puzles, edición de niveles propios.
- Puntos positivos: la mecánica de sigilo está muy bien implementada e integrada con el escenario, los puzles son creativos y los controles del personaje son bastante sensibles.
- Puntos negativos: no existe la posibilidad de compartir los niveles creados por el propio jugador con otras personas.
- Aspecto visual:

Fig. 23. Ejemplo de pantalla de juego para Stealth Inc: A Clone in the Dark (2013)

3.2.4. Dead by Daylight

- Mes de lanzamiento: Junio de 2016
- Género: Acción
- Desarrollador: Behaviour Digital Inc.
- Descripción: en este juego de horror y supervivencia en 3D tenemos la posibilidad de elegir el papel de superviviente o asesino. Dependiendo del rol que elijamos el objetivo será uno u otro. Si tomamos el papel de superviviente, debemos cooperar con otros tres jugadores más para escapar de un recinto cerrado reparando para ello una serie de generadores de energía. Si tomamos el rol de asesino el objetivo será eliminar a tantos jugadores supervivientes como podamos antes de que estos escapen. Es importante remarcar que la perspectiva de los supervivientes es en tercera persona, mientras que la del asesino es en primera persona [24] [25].
- Mecánicas: sigilo, ocultación utilizando elementos del escenario, multijugador, cooperación, gestión de objetos, adquisición de habilidades, roles de personajes.
- Puntos positivos: al disponer de dos roles completamente distintos la experiencia de juego cambia completamente y se enriquece. El aspecto de juego en equipo y cooperación está muy bien logrado. La ambientación en cuanto a los escenarios y efectos de sonido genera la tensión que se busca inicialmente.

- Puntos negativos: disponibilidad un número de mapas bastante limitados y se echa en falta más contenido en el juego. No muy divertido de jugar en solitario, técnicamente discreto para la tecnología actual.

- Aspecto visual:

Fig. 24. Ejemplo de pantalla de juego para Dead by Daylight (2016)

3.2.5. INSIDE

- Mes de lanzamiento: Julio de 2016

- Género: Acción, Aventuras, Indie

- Desarrollador: Playdead

- Descripción: en este juego de plataformas 2D de desplazamiento lateral el jugador controla a un niño en un mundo postapocalíptico, donde deberá hacer uso del ingenio y la lógica para ir resolviendo los puzles que se encontrará en el camino y le permitirán avanzar en el desarrollo de la trama del juego. La variedad de puzles a resolver es inmensa, en algunos habrá que usar la lógica, en otros usar elementos del escenario, en otros la rapidez, etcétera [26] [27].

- Mecánicas: plataformas con navegación lateral en 2D, puzles, interacción con elementos del escenario.

- Puntos positivos: estéticamente está muy trabajado, los efectos de sonido del juego son muy inmersivos, gran variedad de puzles que además son bastante ingeniosos, narrativamente es atractivo para el jugador.

- Puntos negativos: la duración del juego es un poco escasa, unas 5 o 6 horas si el jugador es habilidoso.

- Aspecto visual:

Fig. 25. Ejemplo de pantalla de juego para INSIDE (2016)

3.2.6. Saga de juegos del Profesor Layton

- Mes de lanzamiento: Octubre de 2008 - Agosto de 2018
- Género: Aventuras, Puzle
- Desarrollador: Level 5
- Descripción: en esta saga de juegos de puzles y misterio controlamos al profesor Hershel Layton, que con la ayuda de su aprendiz Luke Triton y su ayudante Emmy Altava, debe resolver una serie de enigmas y misterios que ocurren en diferentes lugares. En cada historia debemos resolver un conjunto de puzles y problemas de lógica para hacer que la trama avance hasta su resolución definitiva [28] [29].
- Mecánicas: puzles, exploración de escenarios, navegación a través de mapas.
- Puntos positivos: gran variedad de puzles con diferentes dificultades, narrativamente son bastante entretenidos, la historia se entrelaza bastante bien junto a la resolución de puzles.
- Puntos negativos: una vez completados los puzles el juego ya no es rejugable.
- Aspecto visual:

Fig. 26. Ejemplo de pantalla de juego para la saga del Profesor Layton (2008 - 2018)

3.3. Especificación de requisitos

Este apartado está dedicado a definir de forma estandarizada la lista de requisitos de la solución propuesta para este trabajo. Generalmente, la fuente principal de los requisitos en la mayoría del software desarrollado hoy en día parte del cliente o el usuario final de la aplicación. Sin embargo, para el presente trabajo y debido a la naturaleza del software desarrollado, tratándose de la creación y diseño de un propio videojuego, se considerará que la fuente principal de los requisitos es el propio autor del trabajo. Este realiza la labor de analista, diseñador y programador del software generado.

La lista de requisitos presentada divide los mismos en dos categorías: requisitos de capacidad y requisitos de restricción. Los requisitos de capacidad sirven para definir qué se quiere crear, o de otra forma, qué funcionalidades va a tener el videojuego. Los requisitos de restricción, por el contrario, sirven para definir cómo se va a construir el videojuego, estableciendo una serie de limitaciones sobre los requisitos de capacidad. Hay que tener presente que el nivel de detalle en la descripción de requisitos que se hace en esta lista no debe ser excesivo, ya que nos encontramos en la fase de análisis. Este nivel de detalle será profundizado en la fase de diseño.

A continuación, se muestra el modelo de tabla que se emplea para la especificación de los diferentes requisitos:

Identificador	**Nombre del requisito**		
Descripción:	Descripción del requisito		
Prioridad:	☐ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☐ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☐ Alta ☐ Media ☐ Baja		

Tabla 2. Modelo de definición de requisitos

Para un mejor entendimiento de la tabla anterior, se procede a describir cada uno de los campos que aparecen:

- Identificador: se trata de un código único mediante el que se puede identificar un requisito. En el caso de los requisitos de capacidad se usará la nomenclatura “RC-XX”, donde “XX” comenzará en “01” y se verá incrementado en uno para cada nuevo requisito de capacidad. En el caso de los requisitos de restricción se usará la nomenclatura “RR-XX”, donde “XX” comenzará en “01” y se verá incrementado en uno para cada nuevo requisito de restricción.

- Nombre del requisito: una o varias palabras que servirán para nombrar a un requisito. Deberá buscarse un nombre coherente con el significado del requisito descrito.

- Descripción: dos o tres líneas máximo donde se describe de forma clara y precisa el objetivo del requisito.
- Prioridad: se trata de una medida de la importancia del requisito para la planificación del proyecto. Su valor es una opción entre "alta", "media" o "baja".
- Fuente: sirve para indicar de dónde se ha extraído el requisito. En todos los casos la fuente será el propio analista.
- Necesidad: mide la importancia del requisito para el futuro jugador. Su valor es una opción entre "esencial", "conveniente" u "opcional". A pesar de contemplar la opcionalidad de alguno de los requisitos, para el presente trabajo se ha optado por no tratar con este tipo de casos, por lo que todos los requisitos mostrados se cumplirán.
- Verificabilidad: mide de alguna forma lo fácil o difícil que es verificar que se cumple dicho requisito en la fase de evaluación por parte de los usuarios de evaluación.

3.3.1. Requisitos de capacidad

A continuación, se presenta la lista de requisitos de capacidad para el videojuego "Run away or shoot":

RC-01	**Pantalla principal**		
Descripción:	El juego dispondrá de una pantalla principal desde la que el jugador podrá comenzar una nueva partida, ver las mejores puntuaciones y consultar los créditos.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 3. Definición del requisito RC-01

RC-02	Dificultad de juego		
Descripción:	El juego mostrará al jugador una pantalla de selección de la dificultad de juego antes de comenzar una nueva partida. Estas dificultades serán: fácil, normal y difícil.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 4. Definición del requisito RC-02

RC-03	Mejores puntuaciones		
Descripción:	El jugador podrá consultar las mejores puntuaciones obtenidas en las diferentes partidas completadas con éxito.		
Prioridad:	☐ Alta ☑ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☐ Esencial ☑ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 5. Definición del requisito RC-03

RC-04	Créditos		
Descripción:	El juego dispondrá de una pantalla de créditos accesible para el jugador donde se mostrarán las personas involucradas en el desarrollo del juego.		
Prioridad:	☐ Alta ☑ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☐ Esencial ☑ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 6. Definición del requisito RC-04

RC-05	Pantalla de juego de dificultad fácil		
Descripción:	Se diseñará una configuración específica de nivel de juego para la dificultad fácil con dos claves a descifrar y tres enemigos.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 7. Definición del requisito RC-05

RC-06	Pantalla de juego de dificultad normal		
Descripción:	Se diseñará una configuración específica de nivel de juego para la dificultad normal con tres claves a descifrar y cuatro enemigos.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 8. Definición del requisito RC-06

RC-07	Pantalla de juego de dificultad difícil		
Descripción:	Se diseñará una configuración específica de nivel de juego para la dificultad difícil con cuatro claves a descifrar y cinco enemigos.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 9. Definición del requisito RC-07

RC-08	Información general del personaje		
Descripción:	En la pantalla de juego se mostrará, en la parte superior, información acerca de la vida restante del personaje, objeto equipado y munición restante.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 10. Definición del requisito RC-08

RC-09	Tiempo restante		
Descripción:	Para la resolución de una partida se dispondrá de un tiempo máximo de 12 minutos, y se mostrará en todo momento en la parte superior de la pantalla de juego el tiempo restante en segundos.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 11. Definición del requisito RC-09

RC-10	Información de claves descifradas		
Descripción:	Se mostrará en la parte superior de la pantalla de juego el número de claves totales del nivel y las claves ya descifradas.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 12. Definición del requisito RC-10

RC-11	Personaje		
Descripción:	El jugador controlará las acciones del personaje principal del juego mediante el uso de teclado y ratón. Estas acciones serán de movimiento, uso de objetos e interacciones con el escenario.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 13. Definición del requisito RC-11

RC-12	Enemigos		
Descripción:	En todos los niveles de juego, se ubicarán una serie de enemigos que dispondrán de un comportamiento inteligente con el objetivo último de perseguir y derrotar al personaje principal.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 14. Definición del requisito RC-12

RC-13	Inventario		
Descripción:	Se ofrecerá al jugador la posibilidad de gestionar el inventario de objetos que tiene durante el transcurso de una partida, y seleccionar qué objeto quiere tener equipado.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 15. Definición del requisito RC-13

RC-14	Recoger objetos		
Descripción:	El jugador recogerá los objetos y la munición colocada en varios lugares del nivel de juego con la acción de moverse por encima de esas regiones del escenario.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 16. Definición del requisito RC-14

RC-15	Descifrar claves		
Descripción:	Se propondrá un puzle que deberá resolver el jugador para conseguir descifrar cada una de las claves, generadas de forma aleatoria, que le permitirá completar la partida con éxito.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 17. Definición del requisito RC-15

RC-16	Pistas al descifrar claves		
Descripción:	En el descifrado de claves, se mostrará al jugador pistas acerca de cuántas cifras están correctamente posicionadas y cuántas forman parte de la clave pero no lo están.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 18. Definición del requisito RC-16

RC-17	Mecánica de combates		
Descripción:	Se dispondrá de una mecánica de combates entre el personaje y los enemigos, mediante el que haciendo uso de las diferentes armas se pueden derrotar a los enemigos.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 19. Definición del requisito RC-17

RC-18	Mecánica de sigilo		
Descripción:	El juego tendrá en cuenta una mecánica de sigilo, mediante la cual el personaje podrá esconderse de los enemigos por el escenario, teniendo en cuenta su alcance y rango visual.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 20. Definición del requisito RC-18

RC-19	Objetos		
Descripción:	Se podrán utilizar los siguientes objetos en el transcurso de una partida: pistola básica, fusil de asalto, bomba de humo, kit de curación y pistola táser.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 21. Definición del requisito RC-19

RC-20	Interacciones con elementos del escenario		
Descripción:	El jugador podrá interactuar con elementos del nivel de juego, como ordenadores y puertas, a través de una acción de teclado.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 22. Definición del requisito RC-20

RC-21	Pantalla de victoria		
Descripción:	Cuando se complete con éxito una partida, se mostrará al jugador una pantalla de victoria en el que se mostrará la puntuación obtenida y se solicitará un nombre para almacenar el resultado.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 23. Definición del requisito RC-21

RC-22	Pantalla de derrota		
Descripción:	Cuando el jugador sea derrotado o se acabe el tiempo para la resolución de la partida, se mostrará al jugador una pantalla de derrota que servirá de transición a la pantalla principal.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 24. Definición del requisito RC-22

RC-23	Pausar partida		
Descripción:	El jugador podrá pausar y reanudar la partida en la pantalla de juego y de gestión del inventario haciendo uso del teclado.		
Prioridad:	☐ Alta ☑ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☐ Esencial ☑ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 25. Definición del requisito RC-23

RC-24	Efectos de sonido y música		
Descripción:	El juego dispondrá de diferentes músicas y efectos de sonido acorde a la pantalla en la que nos encontramos y las acciones que se producen.		
Prioridad:	☐ Alta ☑ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☐ Esencial ☑ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 26. Definición del requisito RC-24

3.3.2. Requisitos de restricción

A continuación, se presenta la lista de requisitos de restricción para el videojuego “Run away or shoot”:

RR-01	Plataforma		
Descripción:	El juego funcionará exclusivamente para plataforma de PC.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 27. Definición del requisito RR-01

RR-02	Controles		
Descripción:	Los controles de juego requerirán del uso de teclado y ratón por parte del jugador.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 28. Definición del requisito RR-02

RR-03	Diseño de interfaces		
Descripción:	Las interfaces de juego se diseñarán de forma lo más intuitiva y lógica para el jugador, teniendo en cuenta tanto el uso de colores como el posicionamiento de los elementos en las pantallas.		
Prioridad:	☐ Alta ☑ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☐ Esencial ☑ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 29. Definición del requisito RR-03

RR-04	Gráficos		
Descripción:	Para el diseño de los escenarios, personajes y enemigos se utilizará un tipo de gráficos "retro" o *pixel art*, manteniendo una coherencia entre todos los elementos.		
Prioridad:	☐ Alta ☑ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☐ Esencial ☑ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 30. Definición del requisito RR-04

RR-05	Game Maker: Studio 2		
Descripción:	El juego será programado haciendo uso de la herramienta de desarrollo Game Maker: Studio 2.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 31. Definición del requisito RR-05

RR-06	Almacenamiento de ficheros		
Descripción:	Se almacenará en el propio PC del jugador un fichero para almacenar información sobre las mejores puntuaciones de las partidas.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 32. Definición del requisito RR-06

RR-07	Metodología de desarrollo		
Descripción:	El juego seguirá una metodología de desarrollo en cascada con las siguientes fases: análisis, diseño, implementación y evaluación.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 33. Definición del requisito RR-07

RR-08	Legislación		
Descripción:	En ningún caso se infringirá algún punto definido en leyes como la de protección de datos o de propiedad intelectual.		
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☑ Esencial ☐ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 34. Definición del requisito RR-08

RR-09	Requisitos del sistema		
Descripción:	Los requisitos del sistema necesarios para poder jugar al juego serán los mismos que los descritos por la herramienta Game Maker: Studio 2.		
Prioridad:	☐ Alta ☑Media ☐ Baja	**Fuente:**	Analista
Necesidad:	☐ Esencial ☑ Conveniente ☐ Opcional		
Verificabilidad:	☑ Alta ☐ Media ☐ Baja		

Tabla 35. Definición del requisito RR-09

4. DISEÑO DE LA SOLUCIÓN

4.1. Diseño del juego

El objetivo de este capítulo de diseño del juego es detallar algunos aspectos referentes al contexto del juego, que no se describieron en detalle en la fase de análisis, como son la sinopsis, los aspectos de jugabilidad y la mentalidad que se busca en el jugador cuando juegue una partida al juego diseñado.

4.1.1. Sinopsis

Juegas como Mel, un vigilante de seguridad que trabaja en unos laboratorios médicos. En tu turno de descanso, te despiertas bruscamente por el ruido de alarmas que advierten de que se ha producido un incidente en las instalaciones. La mayoría de los trabajadores del complejo se han infectado de un virus que los transforma en criaturas hostiles. Debido a los protocolos de seguridad, la única salida de las instalaciones ha sido bloqueada gracias a una serie de ordenadores que se encuentran en la instalación y, además, transcurridos varios minutos los laboratorios serán destruidos.

Tu objetivo es escapar de las instalaciones a tiempo, teniendo para ello que descifrar las claves de los ordenadores que bloquean la salida. Deberás hacer uso de tus habilidades de sigilo y pericia para escapar. Respecto a las criaturas que te acechan… Run away or shoot!

4.1.2. Jugabilidad

“Run away or shoot!” es un juego de acción y puzles desde una perspectiva en tercera persona. El jugador se encuentra atrapado en un complejo donde se encuentran una serie de enemigos, a los que cuales se deberá evitar o neutralizar temporalmente para proseguir con el objetivo final.

En el caso de que el jugador decida enfrentarse a algunos enemigos, dispondrá para ello de algunas armas cuerpo a cuerpo y a distancia, con su munición correspondiente, que se irá encontrando en las instalaciones. Los enemigos no podrán ser derrotados de forma definitiva, debido a que tienen capacidad de regeneración, sino que son neutralizados durante 10 segundos.

En el caso de que el jugador decida centrarse en la faceta del sigilo deberá planificar sus movimientos con respecto a los de los enemigos para evitar ser localizado. También tendrá que tener especial cuidado con el ruido generado por acciones que generan mayor ruido, entre las que están disparar y correr, ya que podrían atraer la atención de enemigos que se encuentren a una cierta distancia.

Los enemigos son humanos infectados que se encuentran vagando por el complejo. Su comportamiento es hostil con el jugador, y le irán a atacar siempre que se encuentre a rango visual. Su velocidad es similar a la del jugador, por lo que en caso de ser detectado sólo le quedará la opción de huir y esconderse o enfrentarse al enemigo si dispone de algún arma y tratar neutralizarlo, ganando algo de tiempo.

Para escapar del complejo el jugador debe encontrar una serie de ordenadores localizados en diferentes salas y desbloquear sus claves. Para realizar esto deberá resolver un puzle en tiempo real consistente en descifrar un código numérico compuesto de 4 cifras con números del 1 al 6 que no se pueden repetir. En cada intento de resolución se ofrece al jugador información acerca de cuántas cifras están bien puestas, y si alguna cifra forma parte del código correcto pero no está en el lugar adecuado.

El objetivo del jugador es escapar del complejo antes de que finalice una cuenta atrás de 12 minutos que comienza a descontarse desde el inicio de la partida.

4.1.3. Mentalidad

En “Run away or shoot!” se busca provocar una mentalidad en el jugador que sea una mezcla de varios sentimientos:

- Tensión y nervios, producidos al tratarse de un juego donde se debe completar una misión antes de un tiempo límite y donde hay una serie de enemigos de los que tendremos que escabullirnos o enfrentarnos en caso extremo.

- Inteligencia y planificación, producidos en la parte de la resolución lógica de los puzles de descifrado de las claves y los aspectos de sigilo y exploración de los laboratorios tratando de no ser detectados por los enemigos.

4.2. Técnica

En este subcapítulo se van a definir algunos de los aspectos técnicos del juego, como son los prototipos de las pantallas, los controles del juego, y de forma más detallada cómo se diseñan las diferentes mecánicas del juego.

4.2.1. Prototipos de pantallas

En este apartado se van a describir cada una de las pantallas que formarán parte del juego. Para cada pantalla se ofrece una descripción lo más detallada posible, tanto de la misma como de los elementos que contiene, teniendo en cuenta la fase de desarrollo en la nos encontramos. Se aportará, a su vez, un prototipo de baja fidelidad respecto a cada pantalla. Se ha optado por este tipo de prototipos debido a que en estas fases del diseño tienen menor coste de desarrollo y son bastante útiles para evaluar los conceptos de diseño del juego.

4.2.1.1. Pantalla de título

Se trata de la primera pantalla que se mostrará al jugador al ejecutar el juego. Esta pantalla sirve de punto de inicio para que el jugador seleccione la acción que desea realizar: comenzar una nueva partida, consultar las mejores puntuaciones registradas o ver los créditos.

A continuación, se muestra un prototipo de baja fidelidad de cómo va a ser la pantalla de título:

Fig. 27. Prototipo de pantalla de título

La pantalla de título constará de los siguientes elementos:

- El título del juego, para el que se optará por un tipo de letra, tamaño y color adecuado para resaltar sobre la imagen de fondo.
- Una imagen de fondo que será acorde a la temática y ambientación del juego, en este caso, se buscará una imagen en la que se represente al personaje principal siendo perseguido por un enemigo.
- Diferentes opciones de selección de lo que desea hacer el jugador. Para navegar entre las diferentes opciones se utilizarán las flechas arriba y abajo del teclado, mostrándose una flecha indicativa como la mostrada en el prototipo para indicar en todo momento qué opción es la elegida. Para confirmar la opción se presionará la tecla de intro. En caso de querer salir del juego se deberá pulsar la tecla de escape.

4.2.1.2. Pantalla de selección de dificultad

Esta pantalla será accedida una vez sea confirmada por el jugador la opción de “Nueva Partida” de la pantalla de título. El objetivo de esta pantalla es permitir al jugador seleccionar en qué nivel de dificultad desea jugar la partida.

A continuación, se muestra un prototipo de cómo va a ser la pantalla de selección de dificultad:

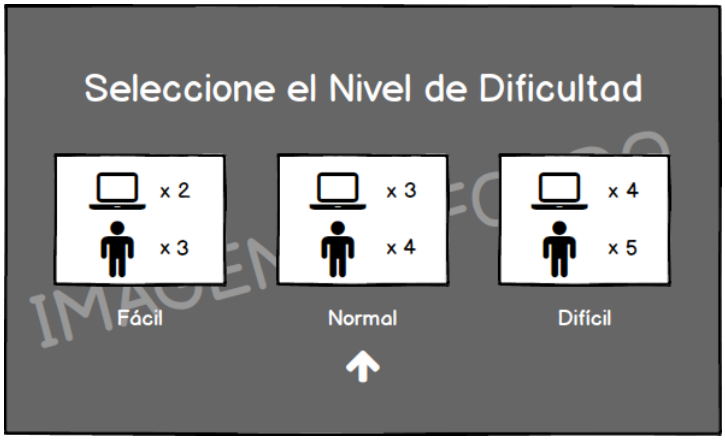


Fig. 28. Prototipo de pantalla de selección de dificultad

La pantalla de selección de nivel de dificultad contendrá los siguientes elementos:

- El título de la pantalla, con un tipo, tamaño y color de letra apropiado para este tipo de pantalla, que resalte sobre la imagen de fondo.
- Una imagen de fondo, que será la misma utilizada en la pantalla de título del juego, para mantener una línea coherente para el jugador.
- Las tres dificultades de juego, de las cuáles el jugador deberá seleccionar y confirmar una de ellas. Estas dificultades tendrán una etiqueta de "fácil", "normal" y "difícil" y una descripción gráfica de las características de cada nivel de dificultad. Cada nivel se diferenciará en el número de claves a descifrar para desbloquear la salida, y en el número de enemigos que se encuentran vagando por el laboratorio. Para navegar entre las diferentes opciones de dificultad se utilizarán las flechas de izquierda y derecha del teclado, mostrándose una flecha indicativa como la mostrada en el prototipo para indicar en todo momento qué opción es la elegida. Para confirmar la opción se presionará la tecla intro. Si se quiere volver a la pantalla de título se utilizará la tecla de escape.

4.2.1.3. Pantalla de mejores puntuaciones

Esta pantalla será accedida una vez sea confirmada por el jugador la opción de "Mejores Puntuaciones" de la pantalla de título. El objetivo de esta pantalla es mostrar al jugador las seis mejores puntuaciones obtenidas en el juego.

A continuación, se muestra un prototipo de bajo nivel de cómo va a ser la pantalla de mejores puntuaciones:

Fig. 29. Prototipo de pantalla de mejores puntuaciones

La pantalla de mejores puntuaciones contendrá los siguientes elementos:

- El título de la pantalla, con un tipo, tamaño y color de letra apropiado para este tipo de pantalla, que resalte sobre la imagen de fondo.
- Una imagen de fondo, que será la misma utilizada en la pantalla de título del juego, para mantener una línea coherente para el jugador.
- La tabla de mejores puntuaciones, en la que se mostrará en la primera columna el nombre del jugador, en la segunda columna el nivel de dificultad en el que se pasó el juego, y en la tercera columna la puntuación final obtenida. La puntuación será de cero en caso de haber sido derrotas, mientras que si se finaliza la partida con éxito es obtenida a partir del tiempo restante de juego en segundos y la dificultad de juego, como se muestra en la siguiente fórmula:

  $$Puntuación = 1 \times Tiempo\ restante\ (en\ seg) + Bonus\ por\ dificultad$$

  dónde el bonus por dificultad es de cero puntos en caso del nivel fácil, cien puntos en caso del nivel normal y de doscientos puntos en caso de nivel difícil. Para salir de esta pantalla el jugador deberá pulsar la tecla de escape.

4.2.1.4. Pantalla de créditos

Esta pantalla será accedida una vez sea confirmada por el jugador la opción de "Créditos" de la pantalla de título. El objetivo de esta pantalla es mostrar al jugador las personas involucradas en el desarrollo del juego.

A continuación, se muestra un prototipo de bajo nivel de cómo va a ser la pantalla de créditos:

Fig. 30. Prototipo de pantalla de créditos

La pantalla de créditos contendrá los siguientes elementos:

- El título de la pantalla, con un tipo, tamaño y color de letra apropiado para este tipo de pantalla, que resalte sobre la imagen de fondo.
- Una imagen de fondo, que será la misma utilizada en la pantalla de título del juego, para mantener una línea coherente para el jugador.
- Un texto de créditos, que permita al jugador poder ver las personas involucradas en el desarrollo del juego. Para salir de esta pantalla de créditos el jugador deberá pulsar la tecla de escape.

4.2.1.5. Pantalla de juego

Esta pantalla será accedida una vez sea confirmada por el jugador uno de los niveles de dificultad de la pantalla de selección de dificultad. Esta pantalla representa una situación normal de partida de juego, cuya estructura será idéntica para cualquier nivel de dificultad.

A continuación, se muestra un prototipo de cómo va a ser una pantalla de juego:

Fig. 31. Prototipo de pantalla de juego

En una pantalla de juego, como la mostrada anteriormente, se pueden distinguir los siguientes elementos:

- La barra de vida, situada en la parte superior izquierda de la pantalla, que servirá para indicar al jugador el total de vida restante de una forma bastante intuitiva.
- Un icono del objeto equipado, situado en segundo lugar de la parte superior de la pantalla, que servirá para indicar qué objeto tiene actualmente equipado el jugador. Más adelante se detallarán qué objetos pueden ser, y de qué forma se cambia el objeto equipado.
- La munición o usos restantes del objeto equipado, situada a continuación en la parte superior de la pantalla.
- Un indicador del número de claves a descifrar completadas y totales del nivel.
- Un contador de tiempo restante expresado en segundos, antes de que el juego se dé por finalizado con derrota.
- El nivel del juego, donde se muestra el total o una porción del mapa en caso de que ocupe más que el tamaño de la pantalla del dispositivo. En la imagen anterior se pueden observar distintas salas, algunas abiertas y otra cerrada. En el caso de la sala cerrada no se puede observar el interior de dicha sala hasta ser abierta, como es lógico, para lo que se presenta con un color por defecto de relleno. En el ejemplo, se pueden ver también cómo podrían ser los enemigos y el personaje principal, así como algunos elementos para proporcionar ocultación al personaje respecto al alcance visual de los enemigos. En algunas de las salas se podrán encontrar los ordenadores, como los que se pueden ver en la imagen, para descifrar las claves y así abrir la puerta de salida de los laboratorios, que también aparece en la imagen representada de color azul y de mayor tamaño que las puertas normales.

4.2.1.6. Pantalla de inventario

Esta pantalla será accedida a partir de la pantalla de nivel, cuando el jugador pulse la tecla “I” del teclado. El objetivo de esta pantalla es permitir al jugador ver los objetos que ha recogido durante el transcurso de la partida y cambiar el objeto que tiene equipado actualmente, si así lo desea.

A continuación, se muestra un prototipo de cómo va a ser la pantalla de inventario:

Fig. 32. Prototipo de pantalla de inventario

En la pantalla de inventario se pueden observar los siguientes elementos:

- Un menú superior, donde se muestra la misma información que la descrita en la pantalla de juego. Se opta por mostrar este menú superior en esta pantalla de inventario debido a que es relevante para el jugador saber la vida restante que tiene, el objeto actualmente equipado, o el tiempo restante. Hay que tener en cuenta, además, que el tiempo sigue contando estando en esta pantalla de inventario. Esto es así, sobre todo, para generar un mayor realismo en la sensación de juego que se quiere producir en el jugador y no permitirle pausar la partida una vez comenzada para planificar los siguientes movimientos.

- El título de la pantalla, con un tipo, tamaño y color de letra apropiado para resaltar sobre el resto de los elementos y que en todo momento el jugador sepa en qué pantalla se encuentra.

- El listado de objetos recogidos durante el transcurso de la partida, para los que se mostrará su imagen representativa, el nombre del objeto y la cantidad de usos restantes o munición para dicho objeto. Para navegar entre los diferentes objetos el jugador deberá usar la tecla de flecha arriba y flecha abajo del teclado, mostrándose de forma bastante intuitiva qué objeto sería el seleccionado. Se sobreentiende que el objeto que quede finalmente seleccionado será el que sustituya al objeto equipado anteriormente.

- Una descripción del objeto seleccionado, en el que se informa al jugador para qué sirve cada objeto y cómo utilizarlo.

4.2.1.7. Pantalla de descifrar clave

Esta pantalla será accedida a partir de la pantalla de nivel, cuando el jugador pulse la tecla de espacio del teclado estando el personaje a una distancia muy cercana frente a un ordenador que aún no haya sido descifrada la clave. El objetivo de esta pantalla es permitir al jugador resolver el puzle correspondiente al descifrado de la clave que le permitirá avanzar en la resolución de la partida.

A continuación, se muestra un prototipo, con un alto grado de fidelidad, de cómo va a ser la pantalla de descifrar clave:

Fig. 33. Prototipo de pantalla de descifrar clave

En la pantalla de descifrar clave, como la mostrada anteriormente, se pueden distinguir los siguientes elementos:

- Un menú superior, donde se muestra la misma información que la descrita en la pantalla de juego. Por el mismo razonamiento que en la pantalla de inventario se decide mantener en esta pantalla de descifrar clave, ya que el tiempo seguirá descontando mientras el jugador intentar averiguar la clave correcta.
- El título de la pantalla, con un tipo, tamaño y color de letra apropiado para resaltar sobre el resto de los elementos y que en todo momento el jugador sepa en qué pantalla se encuentra.
- Una fila de botones etiquetados con los números del "1" al "6" que pueden formar parte de la clave a descifrar.
- Un botón para confirmar la clave, que deberá ser de un tamaño más grande que el resto para resaltar su importancia.
- La clave siguiente a probar por parte del jugador, situada justo debajo de la fila de los botones numéricos.
- Un historial de claves probadas anteriormente, para las que se muestra a su derecha mediante los símbolos de aceptación y de interrogación el número de cifras bien puestas y el número de cifras que forman parte de la clave, pero no están bien puestas.

4.2.1.8. Pantalla de partida finalizada

Esta pantalla será accedida a partir de la pantalla de nivel, cuando el jugador complete con éxito la partida o bien sea derrotado, ya sea porque se haya quedado sin vida debido a ataques de los enemigos o porque el tiempo máximo para completar el nivel llegue a cero segundos. El objetivo de estas pantallas es de información al jugador del resultado final de la partida y de transición de nuevo a la pantalla de título.

A continuación, se muestra un prototipo a bajo de nivel de cómo va a ser las dos pantallas de partida finalizada, en caso de derrota o de victoria:

Fig. 34. Prototipo de pantalla finalizada con derrota

Fig. 35. Prototipo de pantalla finalizada con éxito

Ambas pantallas comparten algunos elementos, descritos a continuación:

- Una imagen de fondo, que será acorde a la ambientación del juego.
- El título de la pantalla, con un tipo, color y tamaño de letra con el fin de resaltar sobre el resto de los elementos y que el jugador sepa en todo momento el resultado final de la partida.
- Un mensaje informativo al jugador para que sepa de qué forma transitar a la pantalla de título.
- Unas imágenes del protagonista y los enemigos, que será diferente dependiendo del resultado del juego. Los enemigos sólo aparecerán en el caso de derrota.

En el caso de la pantalla de victoria, además se incluye, un texto informativo de los puntos obtenidos con la resolución de la partida en el nivel de dificultad concreto y a su vez una caja de texto que el jugador deberá rellenar, si así lo desea, para almacenar su nombre y aparecer en la pantalla de mejores puntuaciones del juego.

4.2.1.9. Mapa de transición de pantallas

Una vez diseñadas y descritas todas las pantallas de las que va a constar el juego, se presenta un mapa de transición entre las diferentes pantallas que serán navegadas mediante las acciones del jugador:

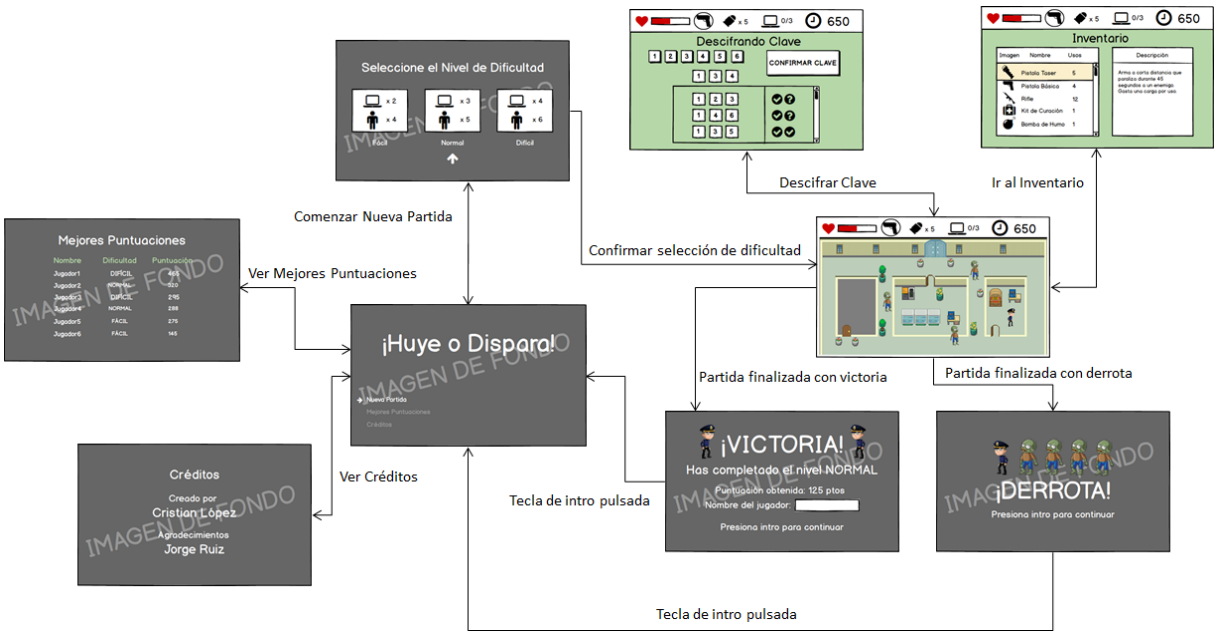
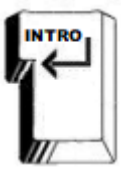


Fig. 36. Mapa de transición de pantallas

4.2.2. Controles

Los controles del juego se plantean para la utilización de teclado y ratón. La idea es que el jugador utilice la mano izquierda para los movimientos del personaje y las interacciones con el entorno con el teclado y la mano derecha para manejar el ratón en acciones de apuntar, disparar y utilizar los objetos equipados.

A continuación, se presenta una tabla con los controles del juego en el que la primera columna se indica la tecla a pulsar y en la columna de la derecha la acción resultante:

	Seleccionar opción de la izquierda en pantalla de selección de dificultad
	Seleccionar opción de la derecha en pantalla de selección de dificultad
	Seleccionar opción de arriba en pantalla de título / pantalla de inventario
	Seleccionar opción de abajo en pantalla de título / pantalla de inventario
	Confirmar opción seleccionada en pantalla de título / pantalla de selección de dificultad / pantalla de partida finalizada

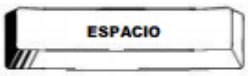
	Mover el personaje hacia arriba
	Mover el personaje hacia la izquierda
	Mover el personaje hacia abajo
	Mover el personaje hacia la derecha
	Ir a la pantalla de inventario / Salir de la pantalla de inventario
	Pausar juego en pantalla de juego
	Abrir puerta / Interactuar con un PC / (Dejando pulsado) Correr
	Volver a la pantalla de título desde pantalla de selección de dificultad / créditos / mejores puntuaciones. Volver al juego desde pantalla de inventario. Salir del juego desde la pantalla de título.
	Disparar / Utilizar objeto equipado / Activar botones de cifras y confirmación en pantalla de descifrar clave

Tabla 36. Controles del juego

4.2.3. Mecánicas

En este apartado se describen de forma más detallada las principales mecánicas en las que se va a basar el juego, con algunos ejemplos ilustrativos para una mayor comprensión.

4.2.3.1. Sigilo

En primer lugar, se tiene la mecánica de sigilo, que tendrá como objetivo la ocultación por parte del jugador respecto a los enemigos que se encuentran por el escenario y la detección del jugador por parte de los enemigos.

Desde el punto de vista del jugador, la ocultación se puede obtener a partir de los diferentes elementos que se encuentran por el escenario. Se considera que todos los elementos son opacos y otorgan ocultación total al personaje si este elemento se

encuentra físicamente entre el jugador y el enemigo. A continuación, se muestra un ejemplo donde el jugador se encontraría oculto respecto a un enemigo:

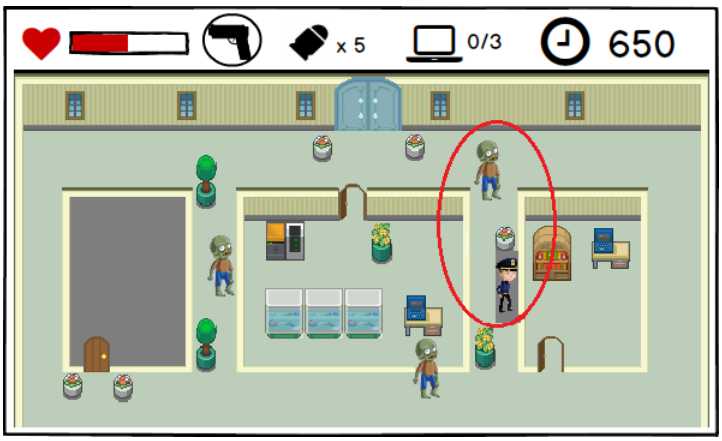


Fig. 37. Ejemplo de ocultación del jugador respecto a un enemigo

Desde el punto de vista de los enemigos, la detección de los jugadores puede lograrse de dos formas. La primera de ellas es mediante el sentido de la vista, si el jugador se encuentra dentro de una línea de visión a una distancia máxima determinada. La segunda vía es mediante el sentido del oído, si el jugador genera algún tipo de ruido (disparos o correr) a una distancia máxima determinada. Ambas situaciones provocarán un cambio de estado en el enemigo que haya detectado al jugador pasando a un estado de hostilidad, que provocará una persecución, mientras siga siendo detectado, y un ataque si se encuentra a una distancia próxima. A continuación, se muestra un ejemplo de detección visual:

Fig. 38. Ejemplo de detección del jugador por parte de un enemigo

4.2.3.2. Recogida de objetos

Una mecánica sencilla que también va a ser implementada en este juego es la recogida de objetos. Existen una serie de armas, munición, y otros objetos de utilidad que se encontrarán en diferentes partes del escenario y el jugador podrá recoger para utilizarlos durante la partida.

La fórmula que se ha optado para su interacción con el jugador es la siguiente: simplemente el jugador tendrá que moverse por encima del objeto para que este sea recogido directamente, sin necesidad de pulsar ninguna tecla. Los objetos recogidos no son directamente equipados, sino que se almacenan en el inventario, donde el jugador podrá gestionarlos y elegir qué objeto equipar y consultar su descripción.

4.2.3.3. Combate

Aunque el juego no está enfocado de forma prioritaria en los combates del personaje con los enemigos, sino más bien en evitarlos, existe una mecánica para este propósito.

La principal regla de los combates, y que es bastante diferencial respecto a otros juegos, es que los enemigos no pueden ser eliminados completamente de la partida, sino que estos sólo podrán ser neutralizados durante 10 segundos. Durante ese tiempo, los enemigos no podrán realizar acciones de mover y atacar. Una vez recuperados, pasarán a actuar de forma habitual, vagando por el mapa y si detectan al jugador pasar a un estado de hostilidad.

En el caso de realizar un ataque de un enemigo al jugador este verá su total de vida reducido en una cuarta parte del máximo de vida, y a continuación el enemigo tardará un tiempo de 2 segundos en reaccionar y continuar moviéndose o atacando.

En el caso de los ataques del jugador a los enemigos, dependiendo del tipo de arma utilizada se aplicará un daño concreto al enemigo atacado, y si su total de vida se reduce a cero pasará al estado de aturdimiento anteriormente descrito.

4.2.3.4. Puzles

La mecánica de los puzles es uno de los elementos más importantes del juego. Se ha diseñado de tal forma que sea lo más fácilmente entendible para el jugador.

Para iniciar un puzle el jugador deberá activar alguno de los ordenadores que se encontrará explorando el escenario. Una vez se muestra la pantalla de descifrar clave, todos los elementos de la partida se paralizarán, excepto el tiempo de finalización de partida, que seguirá disminuyendo a medida que el jugador intenta resolver el puzle.

Cada puzle consiste en descifrar un código numérico compuesto de 4 cifras con números del 1 al 6. En cada intento de resolución se ofrece al jugador pistas acerca de cuántas cifras están bien puestas, y si alguna cifra forma parte del código correcto pero no está en el lugar adecuado. Para esto se utilizan dos iconos diferentes (símbolo de aceptación y símbolo de interrogación) que se mostrarán en la parte derecha del historial de claves introducidas. Es importante también señalar que no hay límite de intentos de resolución de clave. Respecto a las posibles claves, son generadas de forma aleatoria, pero teniendo en cuenta que no puede formar parte de una clave dos cifras iguales.

A continuación, se muestra un ejemplo de pantalla de descifrar clave para un caso en el que la clave es “356”:

Fig. 39. Ejemplo de resolución del puzle de descifrado de clave

4.3. Diseño de niveles

4.3.1. Nivel de juego

Aunque ya ha sido brevemente descrito en la pantalla de juego cómo se va a componer un escenario o nivel de juego, en este apartado se va a pasar a analizar más en detalle los diferentes elementos que se pueden encontrar en un nivel. Pese a existir diferentes dificultades de juego, el tipo de elementos que se van a encontrar en la partida no van a variar, tan sólo su número y distribución. A continuación, se listan los diferentes objetos que pueden aparecer en un nivel de juego en "Run away or shoot!", distribuidos en dos categorías: objetos de ambientación y objetos interactivos.

4.3.1.1. Objetos de ambientación

Se denomina objeto de ambientación a aquel tipo de objeto que se utiliza para construir y dar forma al nivel de juego, pero sobre los que el jugador no puede realizar ninguna acción. En el juego que se está tratando sirven para restringir los posibles caminos que tanto el jugador como los enemigos pueden tomar, aportar ocultación al jugador frente a los enemigos, o con fines meramente estéticos o de ambientación.

- Paredes: se utilizan para la construcción de las diferentes salas que conforman el nivel de juego. Ninguna pared del juego se podrá atravesar tanto por los personajes como por munición, y aportarán ocultación frente a enemigos. A continuación, se muestra un ejemplo de las paredes que se utilizan, y cómo queda formada una sala, en este caso sin incluir una puerta.

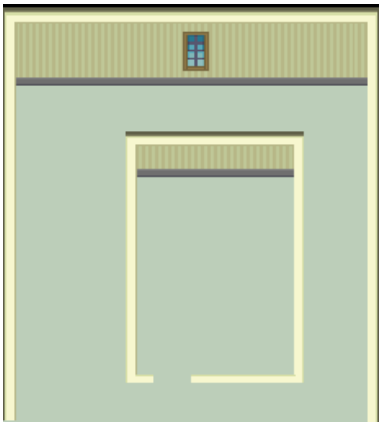


Fig. 40. Ejemplo de paredes en un nivel de juego

- Plantas y arbustos: se utilizan como objeto que aporta ocultación al jugador frente a los enemigos, sobre todo colocados en los pasillos. Los hay de diferente tamaño y apariencia, pero cumplen la misma función. Al igual que las paredes son objetos que no se pueden atravesar. A continuación, se muestran los diferentes tipos de plantas y arbustos que pueden aparecen en un nivel de juego:

Fig. 41. Ejemplo de plantas y arbustos en un nivel de juego

- Mobiliario: cumplen la misma función que las plantas y arbustos descritos anteriormente. La principal diferencia está en que el tamaño de los objetos de mobiliario es superior, y en su estética. Principalmente, se distribuyen en el interior de las salas de las que se compone el nivel. A continuación, se muestran los diferentes objetos de tipo mobiliario que pueden aparecer en un nivel de juego:

Fig. 42. Ejemplo de mobiliario en un nivel de juego

4.3.1.2. Objetos interactivos

Se denomina objeto interactivo a aquel tipo de objeto que puede aparecer en un nivel de juego y con los que el jugador puede relacionarse de alguna forma. Además de aportar ambientación y estética al juego, es con este tipo de objetos con los que se consigue avanzar en el desarrollo de la partida. Los distintos objetos interactivos que se pueden encontrar en un nivel para el juego que se está tratando son:

- Enemigo: personaje controlado mediante inteligencia artificial que se dispone por el nivel de juego. Por defecto, tendrá un comportamiento básico de vagar por un determinado camino de forma repetitiva, que se verá alterado en caso de detectar mediante alguno de sus sentidos, vista y oído, al personaje controlado por el jugador. El otro comportamiento que tiene es de hostilidad hacia el personaje, para lo que le seguirán mientras esté dentro del alcance visual y si está a una distancia muy próxima le efectuará un ataque con un daño concreto. Dependiendo del nivel de dificultad de juego, en el nivel aparecen mayor o menor cantidad de enemigos. Una característica muy importante respecto a los enemigos es su capacidad de regeneración, de forma que su vida vuelve al máximo transcurrido un tiempo después de ser abatidos. A continuación, se muestra qué aspecto tiene un enemigo en el juego:

Fig. 43. Ejemplo de enemigos en un nivel de juego

- Puerta: objeto que sirve de acceso a cada una de las salas que aparecen en un nivel de juego. El jugador podrá interactuar con estos objetos pulsando la tecla espacio para abrir una puerta que esté cerrada y así ver y acceder al interior de la sala asociada a dicha puerta. En un nivel de juego, además, existe un tipo de puerta especial, de mayor tamaño y de color azul, que será la de salida del nivel, y podrá ser abierta una vez se hayan descifrados las claves necesarias. A continuación, se muestra una imagen con las puertas normales, en sus dos estados de abierta o cerrada, y la puerta de salida de nivel.

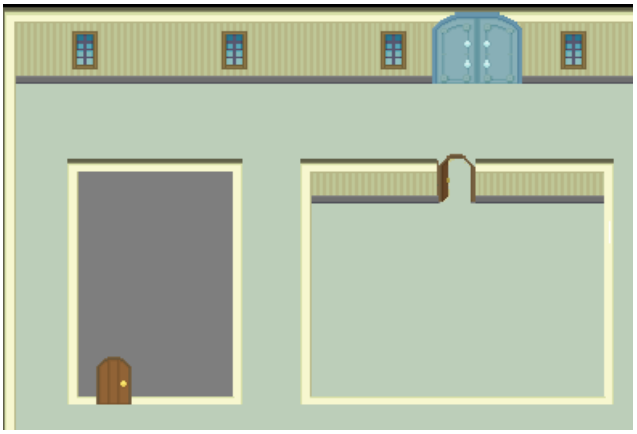


Fig. 44. Ejemplo de puerta abierta y cerrada en un nivel de juego

- Ordenador: objeto que sirve para activar el puzle de descifrar clave si dicha clave no ha sido resuelta ya para ese ordenador. Para ello, el jugador debe pulsar la tecla espacio estando a una distancia muy cercana a un ordenador. Principalmente, se localizan en el interior de las salas del nivel. A continuación, se muestra la apariencia que tiene este tipo de objeto en un nivel de juego:

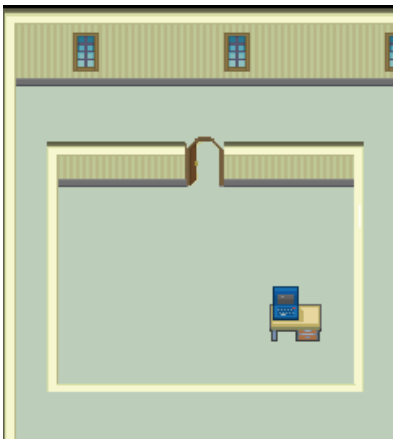


Fig. 45. Ejemplo de ordenador en un nivel de juego

- Ítem/Consumible: cada uno de los objetos de tipo arma, munición o de utilidad que se pueden recoger y utilizar durante la partida. Se disponen en el suelo en diferentes partes del escenario y se recogen de forma instantánea cuando el jugador camina por encima de dichos objetos. Los objetos y/o consumibles que se pueden encontrar en un nivel de juego son: pistola táser, pistola básica, fusil de asalto, kit de curación, bomba de humo y el distinto tipo de munición.

4.3.2. Flujo de juego

En este apartado se va a describir un ejemplo de flujo de juego, desde la perspectiva del jugador, que será muy similar para cualquier tipo de nivel de dificultad. En este ejemplo se describe un fragmento de lo que sería la totalidad de la partida, en el que se consigue descifrar la primera clave del nivel e interactuamos con algunos objetos del juego.

1. Comenzamos en una posición inicial del nivel de juego

2. Caminamos por la sala donde hemos comenzado y recogemos una pistola básica con algo de munición que allí se encontraba
3. Nos dirigimos a la primera sala que nos encontramos con la puerta cerrada, aunque vemos que en el camino transita un enemigo
4. Evitamos al enemigo aprovechando cuando no estamos a su alcance visual y abrimos la puerta de la sala
5. Dentro de la sala recogemos un kit de curación y activamos el primer ordenador del nivel
6. Tras varios intentos, conseguimos descifrar la primera clave del nivel
7. Al salir de la sala, el enemigo que se encontraba merodeando la zona nos ha detectado y se dispone a atacarnos
8. Conseguimos abatir de forma temporal al enemigo haciendo uso del arma que tenemos y salimos corriendo de la zona para dirigirnos a descifrar la siguiente clave del nivel, mientras el tiempo sigue corriendo...

4.4. Diagrama de clases

A partir del diseño de las diferentes pantallas de las que se compone el juego y los diferentes tipos de objetos que se encuentran en un nivel de juego se presenta a continuación el diagrama de clases en UML que se deberá seguir en la fase de implementación del juego:

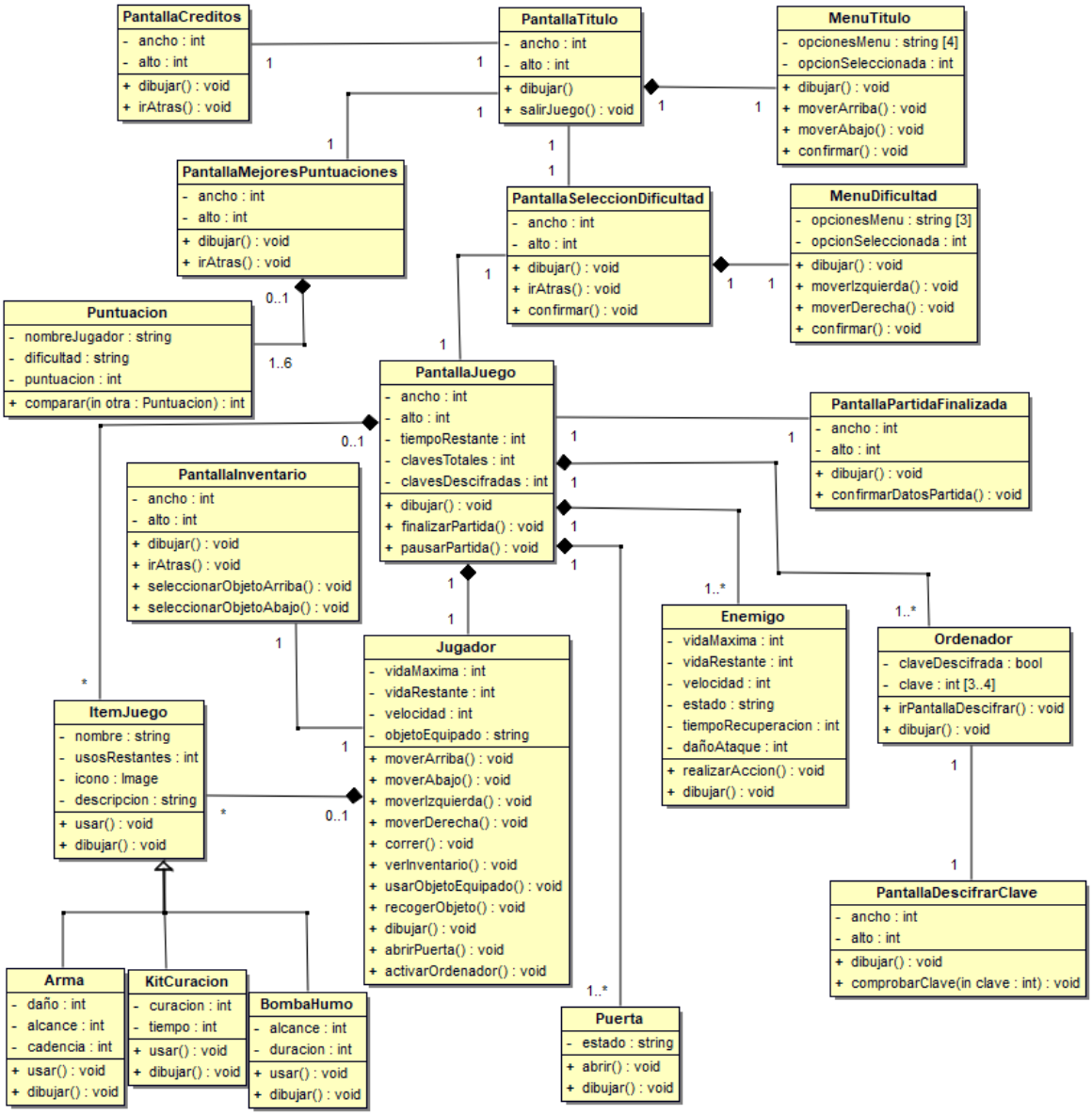


Fig. 46. Diagrama de clases del juego

4.5. Efectos de sonido y música

4.5.1. Estilo

Una parte importante en el diseño del juego en "Run Away or Shoot!" es la elección de la música y efectos de sonido. El principal objetivo es mantener la coherencia con la temática del juego y su ambientación, junto con las sensaciones que se buscan generar en el jugador. Por tanto, el estilo de la música que se va a utilizar en todas las pantallas de juego es música de tensión y que mantenga el suspense.

Respecto a los efectos de sonido a utilizar en las diferentes acciones del juego, no se buscará exagerar los sonidos, sino aportar información útil al jugador de la acción realizada buscando para ello efectos de sonido lo más realistas posibles.

Se deberá tener en cuenta también el tono y el volumen de la música y los efectos de sonido, para no saturar al jugador con estímulos auditivos.

4.5.2. Efectos de sonido

A continuación, se muestra una lista de los diferentes efectos de sonido que se van a utilizar en la construcción del juego, con una breve descripción de cuando se reproducirá dicho efecto:

- Pasos: se reproducirá mientras el jugador esté moviéndose por el escenario, sin correr.
- Pasos rápidos: se reproducirá mientras el jugador esté corriendo por el escenario, semejante al efecto de pasos, pero con una frecuencia de pasos mayor.
- Enemigo detecta a jugador: se reproducirá un tipo de gruñido producido por el enemigo que pasa de un estado normal a un estado de hostilidad tras detectar al jugador. Además de una alerta visual el jugador podrá percibir que se encuentra en peligro.
- Enemigo golpeado: se reproducirá cuando algún enemigo haya sido golpeado por algún arma del jugador.
- Enemigo neutralizado: el jugador podrá saber cuándo el enemigo ha sido neutralizado mediante un efecto a propósito, además de una alerta visual.
- Jugador golpeado: se reproducirá cuando el jugador haya sido golpeado, además de ver su barra de vida reducida así podrá tener este tipo de alerta.
- Jugador derrotado: se reproducirá cuando la vida restante del jugador sea de cero, dando por finalizada la partida y sirviendo de transición a la pantalla de partida finalizada con derrota.
- Disparo con pistola táser: efecto de sonido al utilizar una carga de la pistola táser.
- Disparo con pistola básica: efecto de sonido al utilizar una carga de la pistola básica.

- Disparo con fusil de asalto: efecto de sonido al utilizar una carga del fúsil.
- Utilización de kit de curación: se reproducirá cuando el jugador utilice un kit de curación, además de tener la información visual de ver la barra de vida incrementándose.
- Explosión de bomba de humo: efecto de sonido al explotar una bomba de humo en una zona del nivel.
- Activación de un ordenador: se reproducirá este efecto como alerta al jugador al interactuar con un ordenador, sirviendo de transición a la pantalla de descifrar clave.
- Clave descifrada: el jugador podrá saber cuándo se ha descifrado con éxito una clave acompañando la información visual con este efecto de sonido.
- Intento fallido de resolución de clave: al igual que el efecto anterior, habrá otro para indicar al jugador que la clave no ha sido resuelta con éxito.
- Puerta abriéndose: se reproducirá este efecto como alerta al jugador al interactuar con una puerta cerrada.
- Huir del complejo: este efecto se reproducirá al atravesar el jugador la puerta de salida del nivel, sirviendo de transición a la pantalla de finalización de partida con éxito.

4.5.3. Música

A continuación, se muestra una lista de las diferentes pistas de música que se van a utilizar en la construcción del juego, con una breve descripción de cuando se reproducirá dicha música:

- Música de título de juego: el tema principal del juego que se reproducirá en las pantallas de título, de selección de dificultad, de mejores puntuaciones y de créditos.
- Música de nivel de juego: será la música de fondo a reproducir mientras la partida está en curso, tratando con especial cuidado de no molestar la experiencia de juego del jugador.
- Música de resolución de descifrar clave: música que se reproducirá mientras el jugador está en una pantalla de descifrar clave.
- Música de partida finalizada con éxito: música que se reproducirá en la pantalla de finalización de partida con éxito, escogiendo un estilo más alegre que la música principal del juego.
- Música de partida finalizada con derrota: música que se reproducirá en la pantalla de finalización de partida con derrota, escogiendo un estilo opuesto al anterior.

5. IMPLEMENTACIÓN DE LA SOLUCIÓN

En este capítulo se va a explicar de qué forma se han implementado desde el punto de vista técnico cada uno de los principales aspectos que componen el videojuego creado. Para cada apartado se van a aportar una serie de capturas extraídas de la herramienta Game Maker: Studio 2, intercalándolas con capturas del juego en su versión ya definitiva donde se crea oportuno para ayudar a su comprensión.

5.1. Creación de pantallas

El primer aspecto abordado durante la implementación del videojuego es la creación de las diferentes pantallas que lo componen y para las que se diseñaron prototipos en el capítulo de diseño del juego.

Con el objetivo de no sobrecargar en exceso la presente memoria, y teniendo en cuenta que prácticamente todas las pantallas, a excepción de las de nivel de juego, se implementan siguiendo el mismo esquema, tomaremos como referente la pantalla de título para esta explicación.

El primer paso para construir una pantalla en Game Maker consiste en determinar la estructura de capas de los diferentes elementos que la constituyen. Para la pantalla de título, tal y como se describió en el prototipo diseñado, tendríamos un objeto de título, un menú con tres opciones seleccionables por el jugador, y un fondo.

Estos elementos se incluyen en diferentes capas dentro de la pantalla, que en Game Maker se diferencian entre sí en función del tipo de elementos que se les asocian, ya sean objetos, conjuntos de casillas, fondos y caminos. En concreto, esta pantalla de título se compone de dos capas de instancias y una capa de fondo. La razón de generar dos capas de instancias diferentes es separar por un lado los objetos textuales, título y menú, y por otro lado, los objetos animados. A continuación, se muestra de qué forma se visualiza esta estructura de elementos con la herramienta Game Maker.

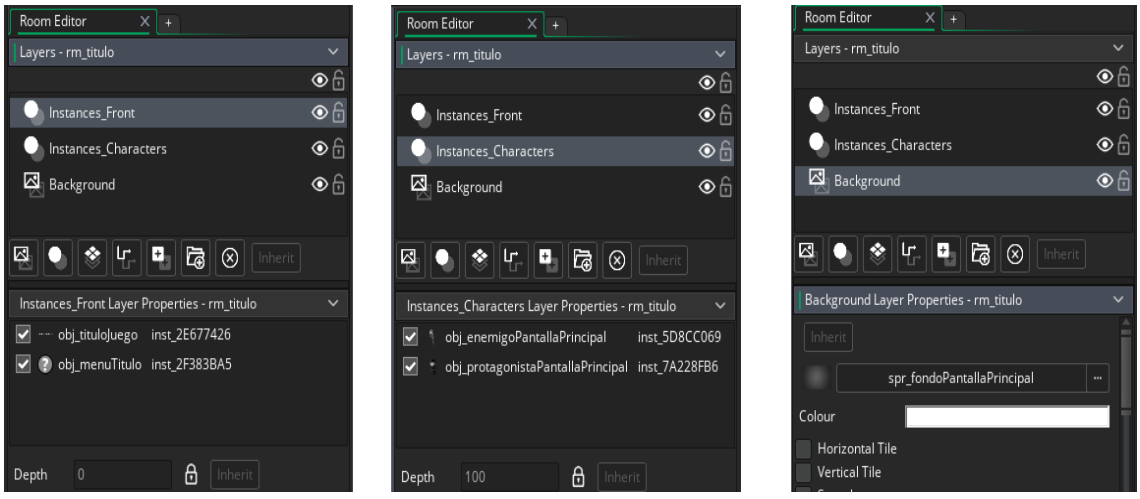


Fig. 47. Estructura de capas de la pantalla de título

La primera capa, denominada *Instances_Front*, se compone de dos instancias de objetos *obj_tituloJuego* y *obj_menuTitulo*. Esta capa tiene un atributo de profundidad (*Depth*)

igual a 0, que siendo el menor de todas las capas de instancias significa que sus elementos son los últimos en dibujarse de todo el conjunto de capas de la pantalla. La segunda capa, denominada *Instances_Characters*, se compone de dos instancias de objetos *obj_enemigoPantallaPrincipal* y *obj_protagonistaPantallaPrincipal*. Esta capa tiene una profundidad mayor que la anterior, ya que los objetos que en ella se encuentran se quieren situar por detrás del menú. Por último, se tiene la capa denominada *Background*, que se crea por defecto en Game Maker cuando se crea una nueva pantalla y su objetivo es dibujar el fondo de pantalla con algún sprite creado previamente para este propósito, en el caso de esta pantalla se denomina *spr_fondoPantallaPrincipal*.

A continuación, se presenta la pantalla de título, tal y como la observa el jugador, en la versión definitiva del juego:

Fig. 48. Pantalla de título

Para programar el comportamiento de los diferentes objetos que componen la pantalla hay que estudiar los tipos de eventos que se pueden producir en la misma por cada objeto. Por ejemplo, en esta pantalla de título tenemos que situar el título del juego en una posición concreta, y esto se realiza a través del evento de creación del objeto, de la forma que se muestra a continuación:

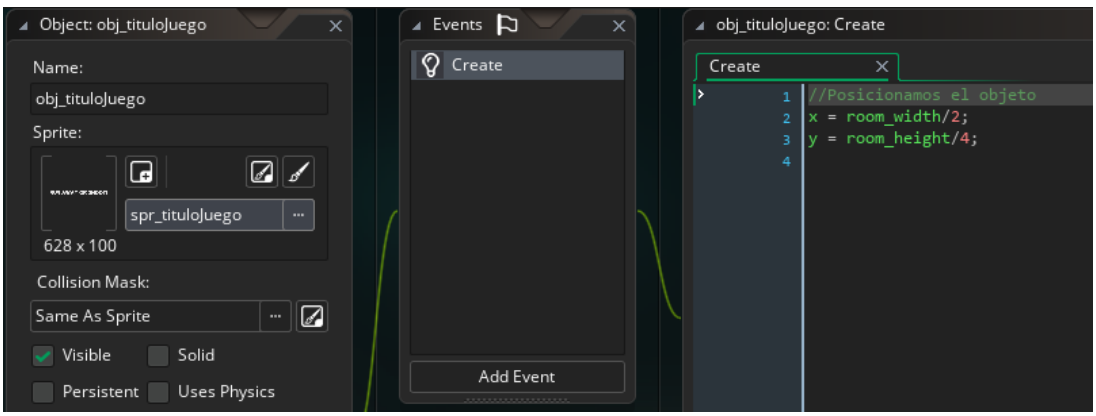


Fig. 49. Programación del evento de creación para obj_tituloJuego

Otro problema a resolver con la implementación de esta pantalla de título sería el comportamiento del menú, y que obedezca a las acciones de teclado del jugador. Para ello se debe programar esta lógica en el evento de tipo paso para el objeto menú, capturando la tecla pulsada y actuando en consecuencia. A continuación, se muestra de qué forma se avanza a la siguiente pantalla en función de la opción seleccionada por el jugador:

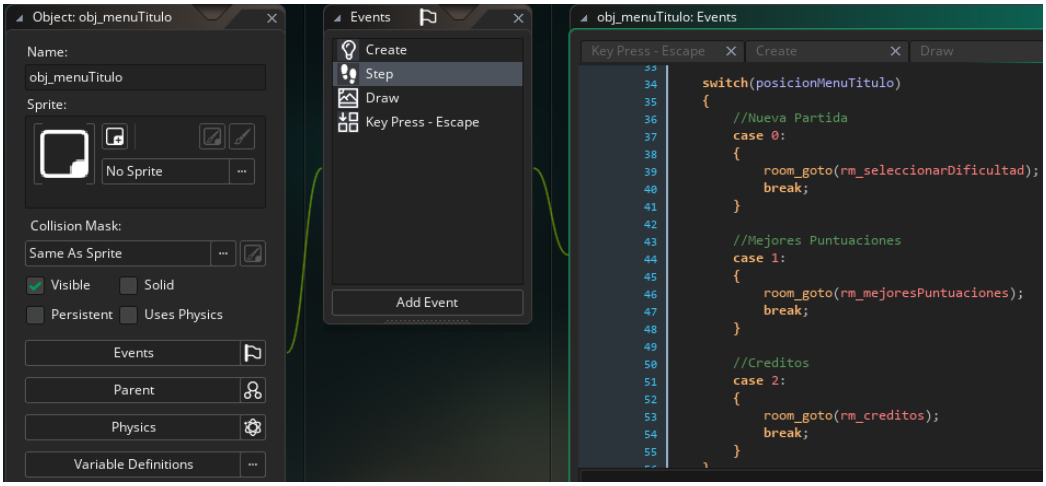


Fig. 50. Programación del evento paso para obj_menuTitulo

5.2. Creación de niveles de juego

Por nivel de juego se entiende a los tres tipos de partidas que se pueden jugar en el juego diseñado atendiendo a la dificultad de juego. Se ha optado por una construcción del escenario de forma exclusiva para cada una de las dificultades, para así aportar una mayor riqueza a la experiencia de juego.

La implementación de este tipo de pantallas es similar a como se ha explicado en el subcapítulo anterior. Sin embargo, existen una serie de diferencias en la implementación de este tipo de pantallas. En primer lugar, la estructura de capas y elementos que las componen es mucho más compleja ya que se disponen de capas destinadas a los caminos que trazarán los enemigos en su comportamiento de patrullaje y un tipo de capa especial dedicada a los conjuntos de casillas que compondrán el escenario de juego.

Es importante señalar que algunas de estas capas no son visibles para el jugador, como, por ejemplo, *InstancesElementosOpacos* o *ComputerInteractionTiles*. Cada una de estas capas invisibles tiene un cometido específico para resolver las distintas mecánicas del juego, como serían en este caso la navegación por el escenario o las interacciones con los ordenadores respectivamente.

Esta estructura de capas tiene que ser consistente en la forma en la que los diferentes objetos se van a ir desplegando sobre la pantalla. Esto significa que, por ejemplo, las instancias de humo generadas cuando explote una bomba de humo deben situarse en una capa más profunda que la capa donde se sitúe la instancia del personaje controlado por el jugador. A continuación, se muestra la estructura de capas de las que se compone el nivel de juego de dificultad fácil:

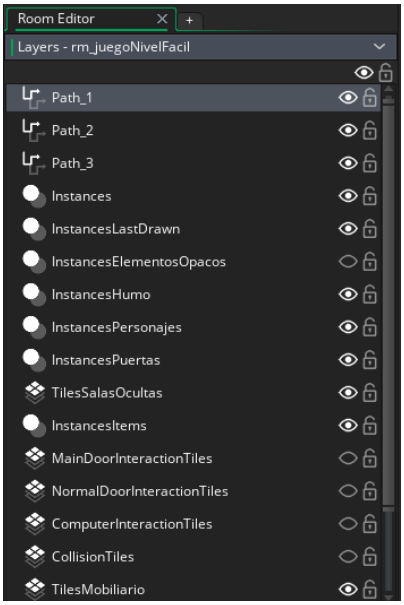


Fig. 51. Estructura de capas del nivel de juego de dificultad fácil

Un aspecto a tener en cuenta en la implementación de este tipo de pantallas es la parte de la vista y cámaras del juego. Es necesario configurar el tamaño de la vista cámara conforme al resto de pantallas del juego, debido a que estas pantallas son de un tamaño mayor. Asimismo, al configurar la cámara hay que establecer de qué forma está se mueve, para lo que se indica que se centre en el objeto del personaje controlado por el jugador. Se puede observar esta configuración en la siguiente imagen:

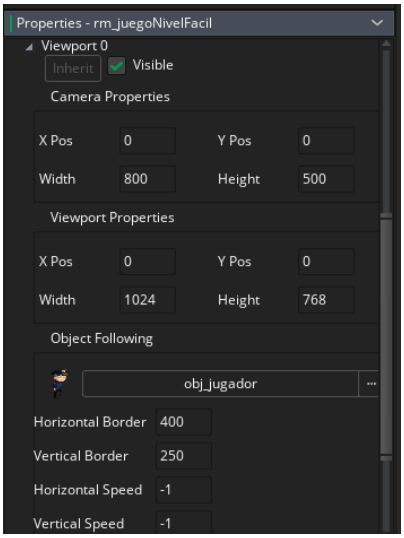


Fig. 52. Configuración de la vista y cámara del nivel de juego de dificultad fácil

Otro elemento novedoso respecto a los otros tipos de pantalla es el de la utilización de los conjuntos de casillas o *tilesets*. Este tipo de objeto en Game Maker sirve para construir los diferentes escenarios de los que se compondrá una pantalla en concreto. Partiendo de una imagen y estableciendo el tamaño del lado de la casilla la propia herramienta nos proporciona un editor bastante intuitivo de construcción de niveles, como se muestra en la siguiente imagen:

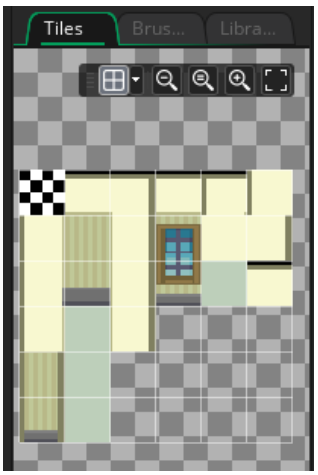


Fig. 53. Constructor de niveles a partir de un conjunto de casillas en Game Maker

Una vez generada la estructura de capas comentada anteriormente y situadas todas las casillas que componen el escenario ya se tendría un nivel de juego creado, desde el punto de vista más externo, ya que la lógica de programación de los diferentes elementos que componen el juego se explicará más adelante en cada uno de los subcapítulos. A continuación, se muestra cómo queda implementado el nivel de juego de dificultad fácil en Game Maker:

Fig. 54. Nivel de juego de dificultad fácil

5.3. Creación de personajes

5.3.1. Protagonista

Una vez implementadas todas las pantallas de las que se compone el juego, y los propios niveles de juego, el siguiente paso es el proceso de implementación del personaje controlado por el jugador.

Una vez creado en Game Maker el objeto correspondiente, que denominamos *obj_jugador*, el primer paso es darle un aspecto visual, o lo que es lo mismo, asignarle

un sprite. A diferencia de otro tipo de objetos que permanecen estáticos durante el transcurso del juego, como el título del juego en la pantalla principal o las casillas, el personaje puede adoptar sprites compuestos de más de una imagen, conformando de esta forma animaciones.

En el caso del protagonista, y en función de los diferentes estados en los que se encuentre o las acciones que esté realizando en un momento dado se tienen los siguientes tipos de animaciones:

- Movimiento: dentro de las animaciones de movimiento se tienen la animación de idle, cuando no hay movimiento, la animación de correr y la animación de caminar.

- Ataque: en las animaciones de ataque se incluyen la animación de ataque con pistola básica, con pistola táser y con fusil de asalto.

- Utilización del kit de curación

Es importante señalar que todas las animaciones tienen dos variantes: una con el protagonista orientado hacia la derecha y la otra con el protagonista orientado hacia la izquierda. Además, hay que señalar que todas las animaciones se componen de al menos cinco frames diferentes, cuyas transiciones se ajustan de acuerdo a una velocidad de entre 10 y 15 frames/segundo.

A continuación, se muestra un ejemplo de cómo se configura en Game Maker una de estas animaciones:

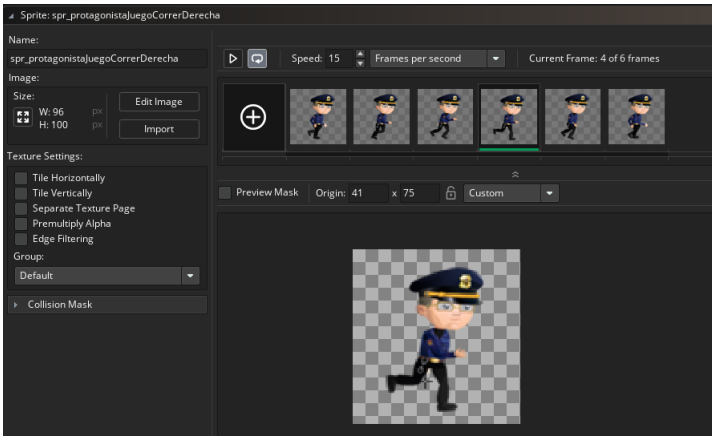


Fig. 55. Configuración de la animación de correr hacia la derecha del protagonista

Un aspecto muy interesante que también se debe configurar del objeto protagonista es la máscara de colisión para cada una de las animaciones que lo componen. La máscara de colisión sirve para determinar el perímetro del objeto y cuándo este se considera que ha colisionado con otro objeto. En el caso del protagonista, se ha optado por definir la máscara de colisión de forma cuadrada ligeramente superior al tamaño de las casillas del escenario, cuyo tamaño es de 20 x 20 pixeles, y centrado en las piernas del personaje. A continuación, se muestra un ejemplo de definición de la máscara de colisión del personaje para la misma animación mostrada anteriormente:

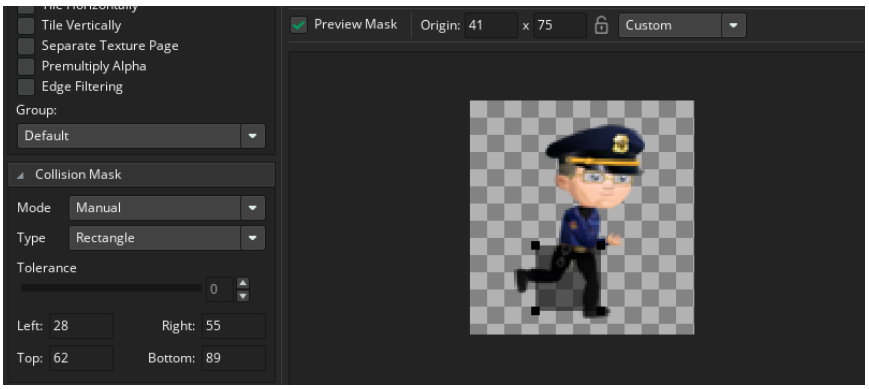


Fig. 56. Definición de la máscara de colisión para una animación del protagonista

Una vez definido el aspecto visual y todas las posibles animaciones que tendrá el protagonista hay que programar tanto su control por parte del jugador como la máquina de estados que controlará estas transiciones entre animaciones y los distintos comportamientos que podrá ir realizando el protagonista.

En el caso del control del jugador se realizará capturando mediante eventos de teclado y ratón las distintas acciones que realice el jugador. Por ejemplo, si el jugador pulsa la tecla "A" se tiene que interpretar un movimiento hacia la izquierda por parte del personaje. Si además, durante dicha acción, se mantiene pulsada la tecla espaciadora hay que interpretar que dicho movimiento tiene que ser corriendo. A continuación, se muestra un ejemplo de qué forma se capturan estas acciones en lenguaje GML:

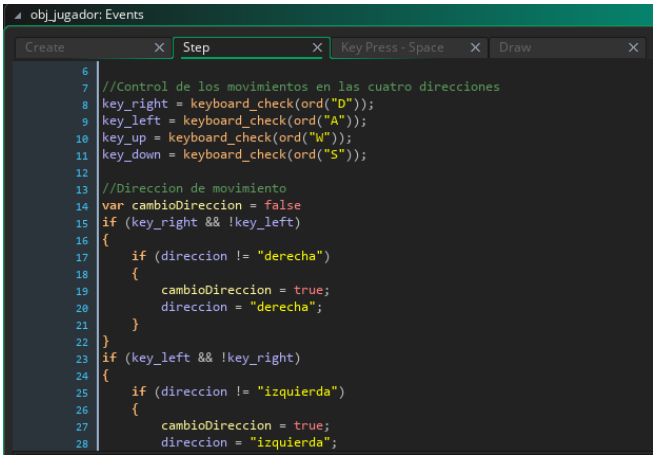


Fig. 57. Programación de la captura de acciones de teclado del jugador

La programación de la máquina de estados del protagonista se realiza siguiendo el siguiente esquema que se repite para cada paso de ejecución del juego:

1- Si el estado actual es de ataque o de curación, la animación actual pasa a ser la correspondiente en función del objeto utilizado, ya sea un arma o el kit de curación, y la orientación del personaje.

2- Si el estado actual no es de ataque o de curación, la animación actual pasa a ser una de las de movimiento en función de las teclas pulsadas por el jugador y las limitaciones de los objetos no traspasables del escenario, como paredes, objetos

de ambientación y puertas cerradas. En este caso las animaciones posibles serían las de idle, caminar y correr.

3- El estado para el siguiente paso de ejecución dependerá de las acciones pulsadas en el ciclo anterior. Por ejemplo, si el jugador pulsa el botón izquierdo del ratón teniendo equipada la pistola básica el siguiente estado será de ataque siempre y cuando tenga munición restante.

A continuación, se muestra un ejemplo de codificación del estado de atacar para el caso de tener equipada la pistola básica:

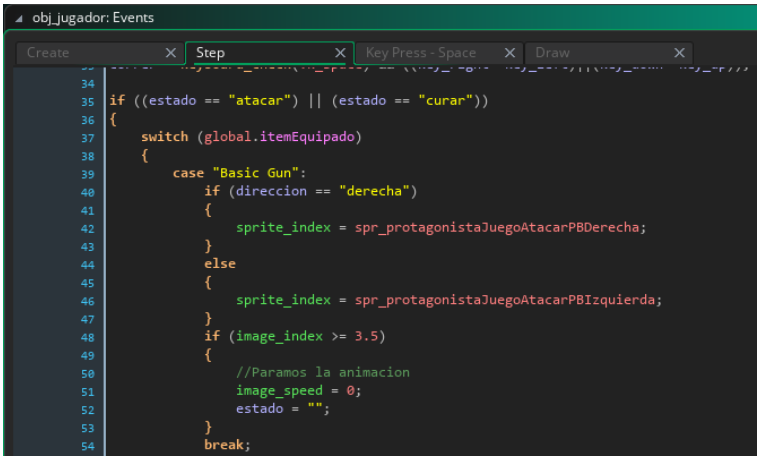


Fig. 58. Programación de la animación para el estado de atacar del jugador

5.3.2. Enemigos

La implementación de los enemigos se ha llevado de una forma similar a la del personaje protagonista, al menos en la parte visual. Dentro de la programación del comportamiento, que se comentará más en detalle en el apartado 5.7 de este mismo capítulo, hay que tener en cuenta que estos personajes serán controlados por la máquina y cuyo objetivo principal es perseguir y derrotar al protagonista.

El objetivo creado para representar a los enemigos se ha denominado *obj_enemigo*, y su aspecto visual puede adoptar una serie de animaciones en función de los estados en los que se encuentre el enemigo en un momento dado. A continuación, se definen las posibles animaciones de un personaje enemigo dentro de Run away or shoot!

- Movimiento: dentro de las animaciones de movimiento se tienen la animación de idle, cuando no hay movimiento, la animación de correr y la animación de caminar.

- Ataque: a diferencia del protagonista los personajes enemigos sólo dispondrán de un tipo de animación de ataque, que será de tipo cuerpo a cuerpo.

- Herido: este tipo de animación se activará cuando el enemigo haya sido abatido de forma temporal, ya que transcurrido el tiempo de la animación volverá a recuperar el total de su salud.

De igual forma que el personaje protagonista todas las animaciones del enemigo tienen sus dos variantes, orientado hacia la izquierda y orientado hacia la derecha. La velocidad de transición de los diferentes frames también está ajustada en todas las animaciones de entre 10 y 15 frames/segundo, para mantener una coherencia con el resto de animaciones del juego y las acciones que se realizan.

A continuación, se muestra un ejemplo de la configuración de la animación de ataque hacia la izquierda del enemigo extraído de Game Maker:

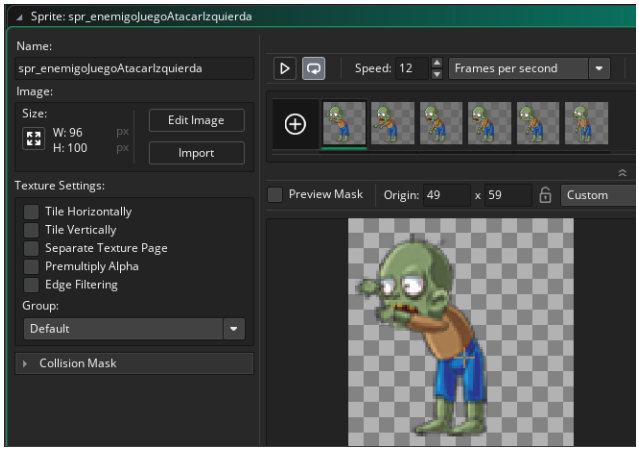


Fig. 59. Configuración de la animación de atacar hacia la izquierda del enemigo

Respecto a la configuración de la máscara de colisión se ha optado también por una de tipo de rectangular, pero a diferencia de la definida para el protagonista, en el caso de los enemigos cubre un mayor área del personaje. Esta decisión se ha tomado para dar un mayor realismo a la mecánica de combate, ya que se quiere considerar como impacto cualquier colisión de un proyectil con cualquier parte del cuerpo del personaje enemigo. A continuación, se muestra de qué forma queda definida la máscara de colisión para los enemigos:

Fig. 60. Definición de la máscara de colisión para una animación del enemigo

Una vez definido el aspecto visual y la máscara de colisión que tienen los enemigos, el siguiente paso de la implementación es la programación del comportamiento. Debido a que se han utilizado técnicas de inteligencia artificial de programación de agentes autónomos se ha decidido tratar con un mayor detalle en el capítulo 5.7.

5.4. Creación de ítems

Dentro de la implementación de los diferentes objetos que se pueden utilizar durante una partida de Run away or shoot!, se tiene que distinguir dos mecánicas bien diferencias: recolección y gestión de ítems mediante el inventario y la mecánica de combate y utilización de objetos de utilidad.

Respecto a la recolección de ítems, cuando se crea un nivel de juego existe una capa de instancias destinada a este tipo de objetos. Durante el evento de creación del nivel de juego se crean manualmente una serie de objetos que serán albergados en esta capa, identificados por un tipo de objeto, una cantidad de munición correspondiente, si corresponde, y una determinada posición dentro del escenario. A continuación, se muestra de qué forma se crean estos objetos dentro del código:

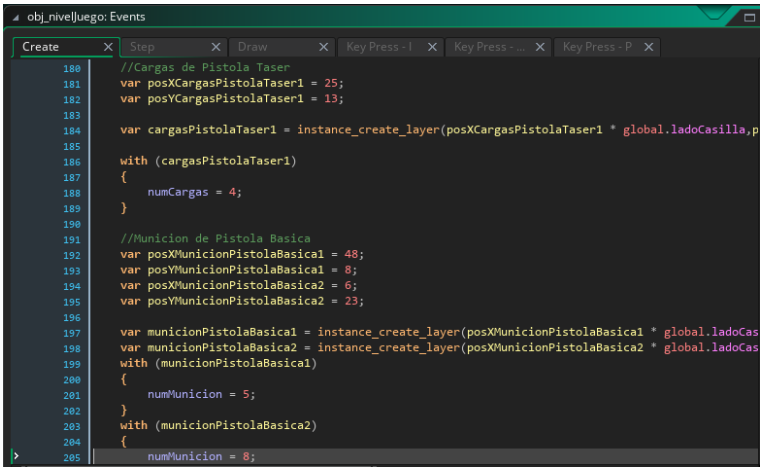


Fig. 61. Programación de la creación de ítems para un nivel de juego

La forma en la que el personaje protagonista interactúa con estos objetos para recolectarlos es mediante eventos de tipo colisión. Cuando el objeto protagonista se mueve por el nivel de juego mediante las acciones del jugador y su máscara de colisión pasa a ocupar parte del área de la máscara de colisión definida para estos objetos recolectables se activa el evento colisión correspondiente. Las acciones resultantes cuando se produce esta colisión es el almacenamiento en el inventario del jugador de la cantidad correspondiente del tipo de objeto colisionado y su posterior eliminación del nivel de juego, como se puede ver a continuación para el caso de la colisión con un objeto de tipo bomba de humo:

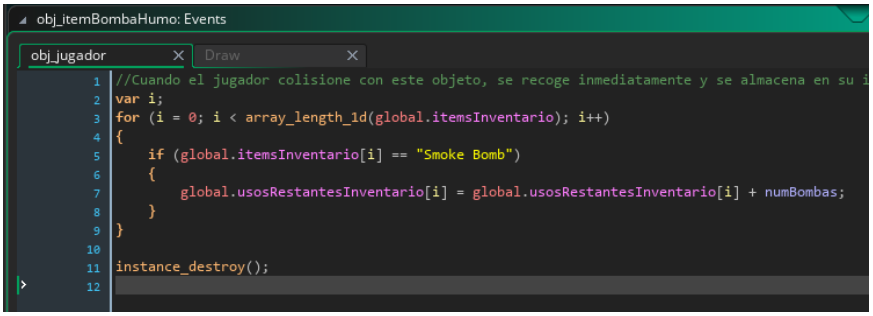


Fig. 62. Programación de la recolección de bombas de humo

La implementación del inventario que posee el jugador en una partida se realiza mediante una estructura de datos de tipo array donde cada posición determina el tipo de objeto en cuestión y el dato guardado es la cantidad de munición o usos restantes para ese objeto. En concreto, los objetos disponibles en una partida son: pistola básica, pistola táser, fusil de asalto, bombas de humo y kit de curación. Para visualizar el inventario el jugador debe pulsar la tecla "I" y se le mostrará una pantalla como la que se muestra a continuación:

Fig. 63. Pantalla de inventario mostrada al jugador

Para cambiar el objeto que tiene equipado en un momento dado, simplemente deberá navegar con las teclas de flecha arriba y flecha abajo. El objeto que se deje seleccionado será automáticamente equipado, para proseguir con la partida.

Cuando el jugador quiere hacer uso de un objeto tiene que hacer click izquierdo en el ratón, independientemente del tipo de objeto que tenga equipado en esos momentos. El único requisito, obviamente, para hacer uso de un objeto es que tenga al menos un uso restante en ese instante. De la misma forma que para otros tipos de eventos de teclado, en Game Maker existe la posibilidad de capturar la acción del click izquierdo del ratón, por lo que ahí se debe programar la mecánica de utilización de objetos.

En el caso de las armas de fuego, su programación es muy similar. La principal diferencia está en el tipo de daño y cantidad de daño que ocasionan a los enemigos, el alcance de disparo, o la cadencia de disparo. Por ejemplo, la pistola básica dispara balas normales a una mayor distancia y quita un 20% del total de vida del enemigo, mientras que la pistola táser tiene un menor alcance y dispara un rayo eléctrico que en lugar de quitar salud del enemigo, lo paraliza durante unos segundos. Para comprobar si se produce un impacto o no en los enemigos cuando se activa un arma, se crea un objeto específico, *obj_bala* en el caso de la pistola básica, con una cierta velocidad constante y dirección. Este objeto tiene su propia máscara de colisión y si pasa a ocupar en algún momento de su existencia dentro del nivel de juego parte del área de la máscara de colisión de un enemigo se produce un impacto. Una vez producido el impacto, se aplica el daño correspondiente en el enemigo y el *obj_bala* se destruye, como se muestra en el siguiente ejemplo de código:

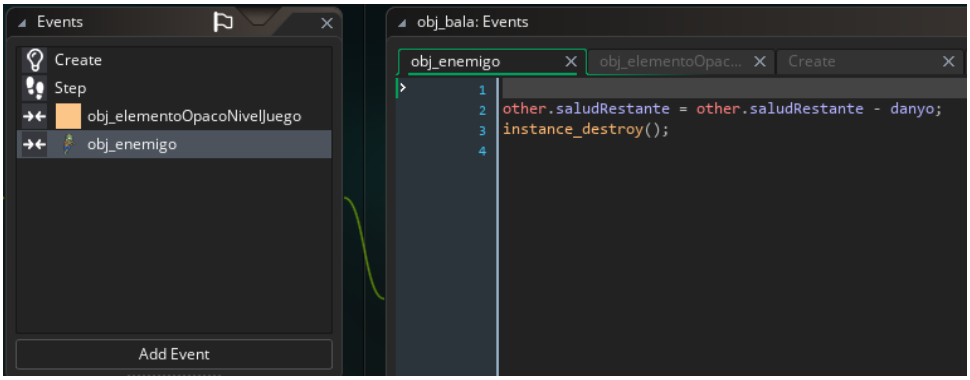


Fig. 64. Programación del evento colisión de una bala con un enemigo

En el caso de los objetos de utilidad, entre los que se encuentran el kit de curación y la bomba de humo, su programación difiere bastante entre sí. La implementación del kit de curación es más sencilla, ya que simplemente se aplica un cambio de estado en el personaje protagonista, pasando a un estado denominado "curar". Debido a que la aplicación del kit de curación transcurre durante unos determinados segundos se activa un contador de tiempo que en cada paso de ejecución incrementa la salud restante del protagonista y una vez que finaliza este contador se pasa a un estado normal de "idle".

La implementación de la bomba de humo es algo más compleja, ya que en primer lugar implica la creación de un objeto denominado *obj_humo* que influye en la visión de los enemigos. Este objeto tiene una determinada forma y máscara de colisión que se comporta como cualquier otro obstáculo para la visión de un enemigo. Sin embargo, tanto el protagonista como los enemigos se pueden mover al interior de una región ocupada por el humo. A continuación, se muestra un ejemplo de partida donde se ha utilizado una bomba de humo y el protagonista se ha ocultado en su interior, pasando inadvertido para los enemigos que pasan por la zona.

Fig. 65. Ejemplo de ocultación del jugador tras una bomba de humo

5.5. Minijuego de descifrar clave

Una de las mecánicas implementadas del juego que añaden algo más de complejidad y alejan de la monotonía la experiencia de juego del jugador es el minijuego del descifrado de claves para escapar del complejo.

Las características del minijuego se definieron completamente en el capítulo de diseño del presente documento, y como tal, se han cumplido en su totalidad en la fase de implementación.

El minijuego de descifrado de claves se corresponde con una pantalla concreta en Game Maker, nombrada como *rm_descifrarClave*. La transición a esta pantalla se produce cuando el jugador interactúa con alguno de los ordenadores localizados en el nivel de juego pulsando la tecla espaciadora. Como cada clave está asociada a un ordenador diferente, es necesario controlar qué ordenador tiene la clave descifrada y cuál no, ya que no se permite resolver más de una clave por ordenador. En caso de que el jugador interactúe con un ordenador y se cumpla la restricción anterior, ya se transita a la pantalla de descifrado de claves, como se muestra a en el siguiente fragmento de código:

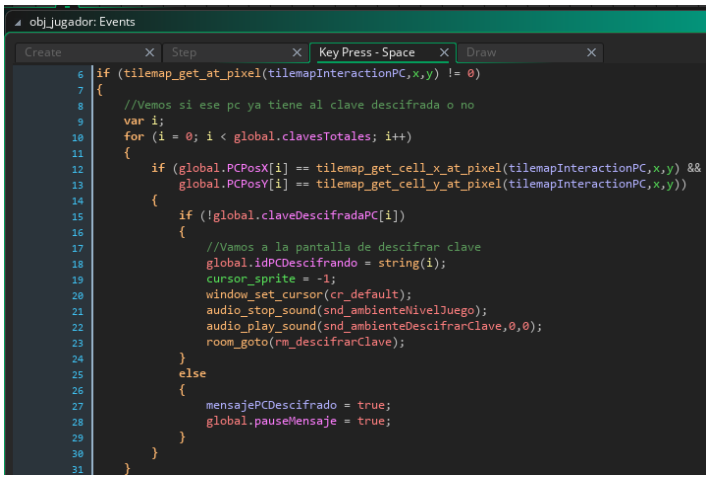


Fig. 66. Programación de la transición al minijuego de descifrar clave tras interactuar con un pc

El primer paso cuando se transita a esta pantalla es la generación de forma aleatoria de la clave a buscar por el jugador. La clave debe estar formada por 6 dígitos del 1 al 6 de forma que ningún dígito aparezca más de una vez en la clave resultante. En Game Maker, existe una función, similar a las que existen en otros lenguajes de programación, que genera un número aleatorio decimal comprendido entre 0 y el número pasado por parámetro. A continuación, se muestra de qué forma se han generado las claves aleatorias:

obj_descifrarClave: Events

Create | Draw | Step | Key Press - P

```
//Generamos la clave de forma aleatoria
randomize();
clave[0] = -1;
clave[1] = -1;
clave[2] = -1;
clave[3] = -1;
var i;
for (i = 0; i < array_length_1d(clave); i++)
{
    var r = random(6);
    var valor = floor(r) + 1;
    if (valor != clave[0] && valor != clave[1] && valor != clave[2] && valor != clave[3])
    {
        clave[i] = valor;
    }
    else
    {
        i--;
    }
}
```

Fig. 67. Generación de claves aleatorias para el minijuego

Una vez que la clave ha sido generada, se almacena en una estructura de tipo array donde en cada posición se almacena de forma ordenada un dígito de la clave que deberá buscar el jugador. La pantalla que se muestra al jugador, finalmente, tiene el siguiente aspecto:

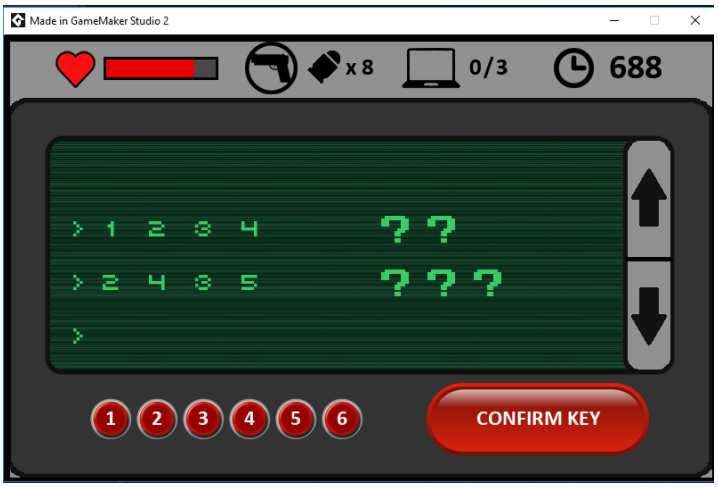


Fig. 68. Ejemplo de pantalla de descifrar clave

Respecto al prototipo propuesto en el diseño, se optó por disponer de otra forma los distintos botones para que resultase más intuitivo para el jugador y su diseño se asemejase a un terminal de computadora antigua. En la parte derecha de la pantalla se dispone de los botones de navegación para visualizar claves introducidas anteriormente. Para disponer de esta funcionalidad es necesario almacenar de forma temporal cada una de las claves introducidas por el jugador en una estructura de datos, así como las pistas asociadas a esas claves devueltas por el sistema. Para facilitar la labor del jugador, los colores de los botones ya pulsados cambian a una tonalidad más oscura y cuando los cuatro dígitos de la clave a probar se han introducido el botón de confirmación se vuelve de color verde.

A continuación, se muestra en qué parte del código se calculan las pistas que devuelve el sistema al jugador cuando este confirma una clave:

```
Draw | Step | Key Press - P

//Vemos cuantas cifras estan bien puestas y cuales aparecen pero no en esa posicion
audio_play_sound(snd_claveIncorrecta,0,0);
var i = 0;
var checks = 0;
var preguntas = 0;
for (i = 0; i < array_length_1d(claveIntroducida); i++)
{
    if (claveIntroducida[i] == clave[i])
    {
        checks++;
    }
    else
    {
        var j = 0;
        for (j = 0; j < array_length_1d(clave); j++)
        {
            if (i != j && claveIntroducida[i] == clave[j])
            {
                preguntas++;
            }
        }
    }
}
historialChecks[clavesProbadas] = checks;
historialPreguntas[clavesProbadas] = preguntas;
clavesProbadas++;
```

Fig. 69. Programación del cálculo de pistas devueltas por el sistema tras confirmar clave

5.6. Almacenamiento de datos

Cuando comienza o finaliza una sesión de juego de "Run away or shoot!", los únicos datos que guardan persistencia, por decisiones de diseño del juego, son los datos de mejores puntuaciones de partidas finalizadas con éxito.

El proceso de carga de los datos se produce siempre al iniciar el juego, a partir de un fichero de texto plano cuyo nombre es *highScoresBeta2.txt*. En el caso de que sea la primera vez que se ejecuta el juego en un determinado ordenador, este fichero no existirá, por lo que se crea con una serie de valores por defecto, como se muestra a continuación:

```
obj_menuTitulo: Events
Key Press - Escape | Create | Draw | Step

//Si no hay un fichero de mejores puntuaciones lo creamos con unos datos por defecto
if (!file_exists(global.nombreFicheroMejoresPuntuaciones))
{
    var fichero = file_text_open_write(global.nombreFicheroMejoresPuntuaciones);
    file_text_write_string(fichero,"Player1");
    file_text_writeln(fichero);
    file_text_write_string(fichero,"Hard");
    file_text_writeln(fichero);
    file_text_write_string(fichero,"465");
    file_text_writeln(fichero);
    file_text_write_string(fichero,"Player2");
    file_text_writeln(fichero);
    file_text_write_string(fichero,"Normal");
    file_text_writeln(fichero);
    file_text_write_string(fichero,"320");
    file_text_writeln(fichero);
    file_text_write_string(fichero,"Player3");
    file_text_writeln(fichero);
    file_text_write_string(fichero,"Hard");
    file_text_writeln(fichero);
    file_text_write_string(fichero,"295");
    file_text_writeln(fichero);
    file_text_write_string(fichero,"Player4");
    file_text_writeln(fichero);
```

Fig. 70. Programación de la escritura de valores por defecto en el fichero de mejores puntuaciones

Como se puede observar, cada tupla de datos de una partida almacenada en el fichero se compone de tres campos: nombre del jugador, nivel de dificultad de juego y puntuación obtenida. Además, para una mayor facilidad en el manejo del fichero, cada campo se almacena en una línea separada.

Una vez generado el fichero de mejores puntuaciones, el siguiente paso es cargarlo en memoria. Para ello se realiza una lectura del fichero una vez se transita a la pantalla de

mejores puntuaciones desde la pantalla de título, en caso de que el fichero no haya sido leído previamente. Los datos leídos se van cargando a estructuras de tipo array, una para cada campo de datos que se quiere manejar. A continuación, se muestra de qué forma se realiza dicho proceso de lectura y carga de datos:

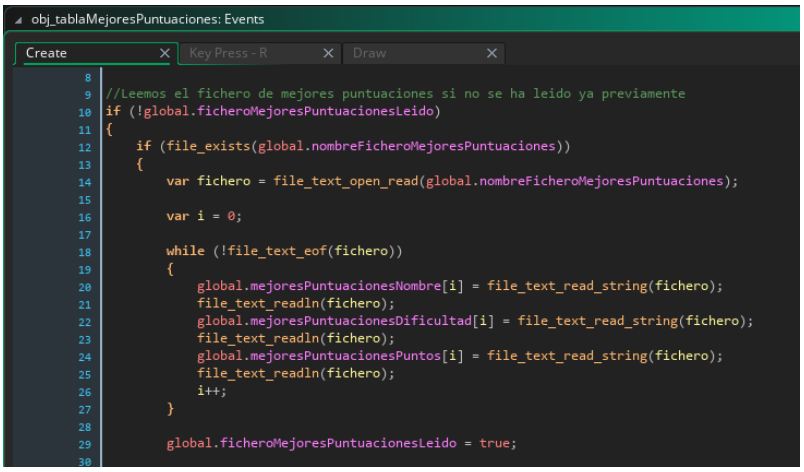


Fig. 71. Programación de la lectura y carga de datos del fichero de mejores puntuaciones

La última operación que se realiza sobre este fichero de mejores puntuaciones es la escritura. Durante una sesión de juego, se irán completando partidas con resultados exitosos o fallidos. En el caso de los resultados exitosos, se obtiene una determinada puntuación calculada de acuerdo a la función descrita en el capítulo de diseño del juego. En el caso de que esta puntuación se encuentre entre las seis mejores puntuaciones hasta el momento, se almacenará en la posición correcta dentro de las estructuras de datos que se comentó anteriormente. Finalmente, cuando se desee salir del juego, pulsando para ello la tecla de escape desde la pantalla de título, se volcarán estas estructuras al fichero de mejores puntuaciones, como se muestra a continuación:

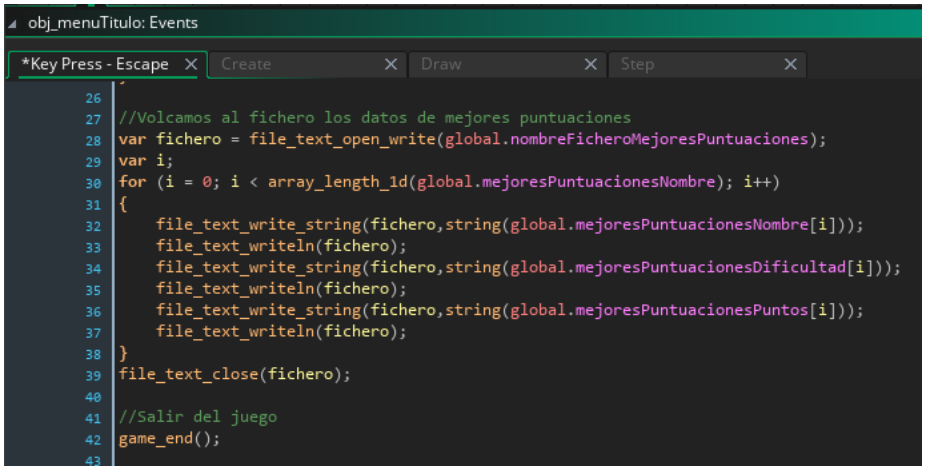


Fig. 72. Programación de la escritura de datos de mejores puntuaciones al fichero

5.7. Programación de la IA de los enemigos

Como se comentó al principio de esta memoria, uno de los objetivos de este trabajo es la utilización de técnicas de inteligencia artificial en alguno de los elementos que componen el videojuego. En concreto, se han utilizado estas técnicas en la programación del comportamiento de los enemigos, tratados como agentes inteligentes en este trabajo.

Se define como agente inteligente a una entidad hardware y/o software con algún grado de autonomía con capacidad para la toma de decisiones y llevar a cabo acciones [30].

Dentro de la construcción de agentes inteligentes se distinguen tres tipos de paradigmas en función de la forma en que los agentes extraen información del medio y actúan en consecuencia:

- Deliberativo: se trata del paradigma más antiguo, en el que se utiliza un modelo del mundo y centrado en la planificación de tareas. El comportamiento de un agente deliberativo es cíclico y siempre se repiten las mismas tareas en el mismo orden: observar el medio, realizar un plan de actuación y ejecutar las acciones.

- Reactivo: este paradigma trata de simplificar la construcción del plan que se lleva a cabo en el paradigma deliberativo. Surge de la idea de construir comportamientos complejos a partir de la agregación de comportamientos sencillos. En este caso directamente las acciones se llevan a cabo a partir de la información recibida por los sensores de los agentes, sin necesidad de una planificación.

- Híbrido: se trata de un paradigma más refinado, que fusiona de algún modo las características de los paradigmas anteriores. Un agente híbrido se caracteriza por realizar una planificación de las acciones a realizar para lograr un determinado objetivo a partir del estado del medio en un momento dado, pero que, sin embargo, es capaz de reaccionar a situaciones imprevistas anulando por completo el plan establecido, de forma totalmente reactiva. De esta forma, se consiguen agentes más versátiles que en los paradigmas anteriores y qué muestran un comportamiento más inteligente si cabe.

En el caso de los enemigos se ha optado por utilizar el paradigma híbrido, con las siguientes características:

- El comportamiento por defecto de un agente enemigo es el de patrullar un cierto área del nivel de juego, siguiendo para ello un camino preestablecido.

- Este comportamiento se altera en función de la información extraída por dos sentidos que posee el agente enemigo: la vista y el oído.

- El sentido de la vista proporciona al agente la información acerca de la distancia entre dicho agente y el personaje protagonista. En el caso de que esta distancia sea menor que un valor de 300 pixeles y no exista ningún objeto opaco entre medias de ambos personajes se considera que el enemigo tiene a la vista al protagonista, para lo que cambia su comportamiento a un estado de persecución.

- El sentido del oído proporciona al agente la información de la generación de ruidos por parte del personaje protagonista. Cuando el protagonista realiza alguna acción que produzca ruido, como puede ser disparar un arma de fuego, detonar una bomba de humo, o correr, se genera ruido. Si el agente enemigo se encuentra dentro de una distancia menor de 150 pixeles se considera que el enemigo tiene a alcance auditivo al protagonista, para lo que cambia su comportamiento a un estado de persecución.

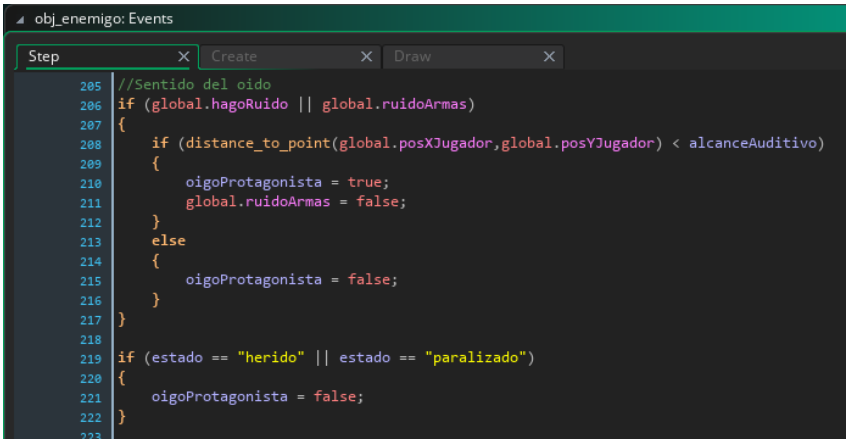


Fig. 73. Programación del sentido del oído del agente enemigo

- Una vez que la parte reactiva del agente enemigo activa el estado de persecución, entra en juego la parte deliberativa del agente: la planificación del camino a recorrer entre el agente y el protagonista o la fuente de ruido. La implementación de esta parte se realiza a partir de funciones que proporciona la propia herramienta Game Maker para la búsqueda de caminos más cortos entre dos objetos que se encuentren en un nivel de juego.

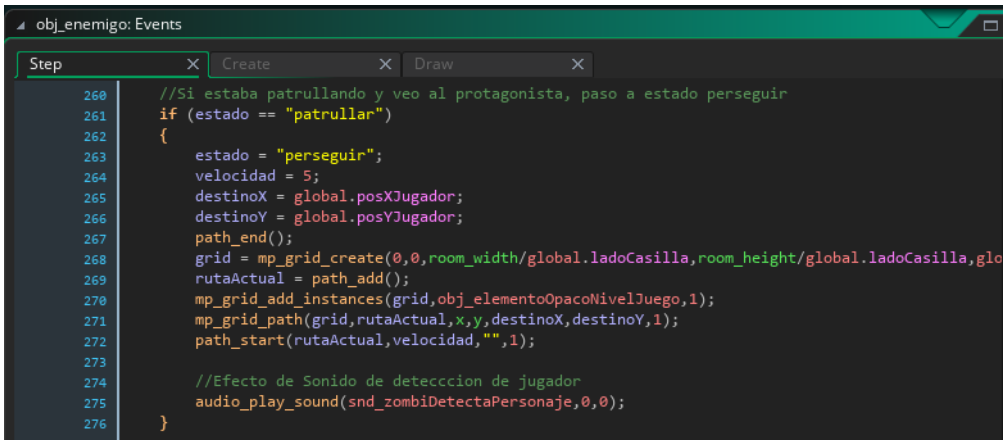


Fig. 74. Programación de la planificación de caminos en estado de persecución

- Para controlar el comportamiento de los enemigos y las diferentes transiciones se incluye un estado, que para el personaje enemigo puede tomar los valores de “patrullar”, “regresar”, “observar”, “perseguir”, “atacar”, “paralizado” y “herido”.

A continuación, se presenta una figura que muestra el esquema de la arquitectura híbrida implementada finalmente en un agente enemigo, pudiendo observar de qué forma interactúan entre sí los elementos internos y externos al propio agente:

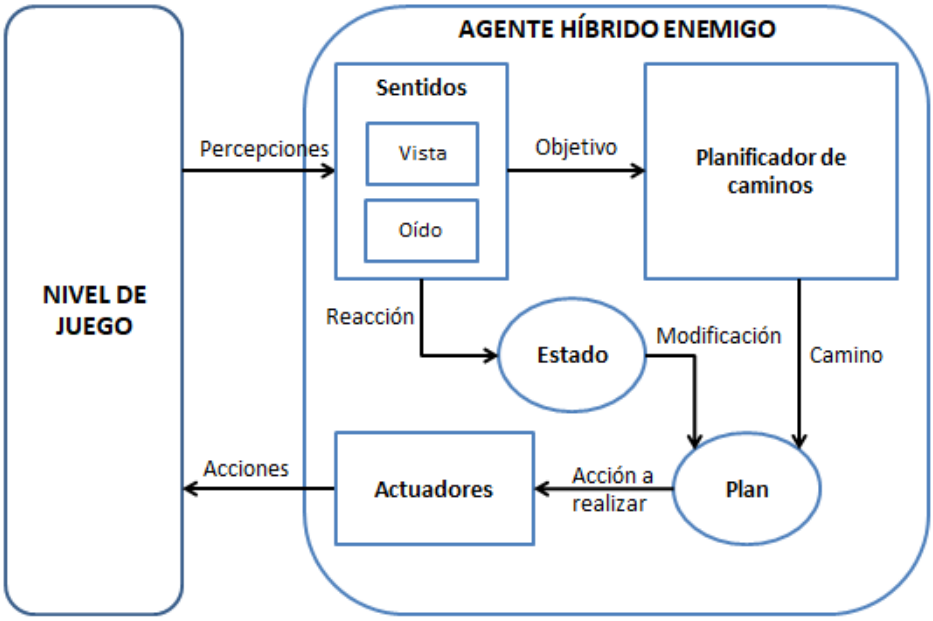


Fig. 75. Arquitectura híbrida del agente inteligente enemigo

5.8. Efectos de sonido y música

El último elemento del juego que se ha implementado son los efectos de sonido de las diferentes acciones que se producen durante una partida y la música que aparece en cada una de las pantallas que componen el juego.

Por razones económicas y teniendo en cuenta el contexto académico en el que se engloba este trabajo las distintas pistas musicales y efectos de sonido utilizados han sido descargados de páginas web de índole colaborativas como *freesounds.org* [31], cuya finalidad principal es la de ofrecer una gran base de datos de sonidos que se puedan utilizar en campos como la investigación científica, o ser integradas en aplicaciones sin uso comercial, o al menos sin tener que pagar una licencia para ello.

Tantos los efectos de sonido como las pistas musicales que se reproducen en el juego se implementan de una forma muy similar en Game Maker. Para explicar el proceso de implementación de un efecto de sonido se tomará como ejemplo el caso del sonido producido al correr el personaje protagonista.

El primer paso es crear un recurso de tipo *Sounds*, y darle un nombre determinado a dicho recurso, que en este caso será *snd_pasosFuertes*. Una vez se crea el recurso, en Game Maker aparece la siguiente pantalla donde, además de seleccionar el fichero fuente donde se localiza el sonido a reproducir, se pueden configurar una serie de atributos como el volumen o la calidad del sonido.

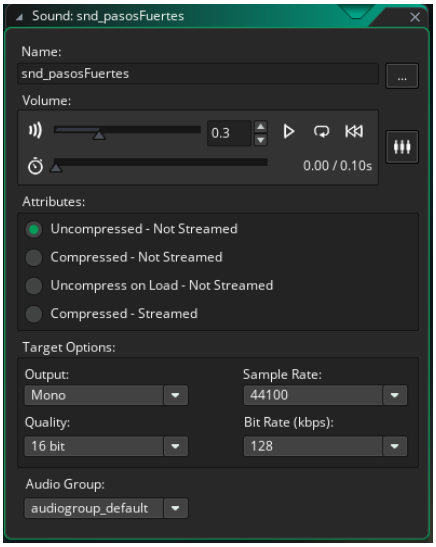


Fig. 76. Configuración de un recurso de sonido

Una vez creado y configurado convenientemente el recurso de sonido, hay que localizar en qué parte del código del juego se activa el evento que produce dicho sonido. En este caso, la acción que ocasiona la reproducción de este efecto de sonido es la acción de correr por parte del personaje protagonista. Por lo tanto, tenemos que averiguar en qué parte del código se activa el estado de correr del protagonista y reproducir este sonido haciendo uso de las funciones que proporciona la propia herramienta Game Maker. A continuación, se muestra en qué parte concreta del código se reproduce este sonido:

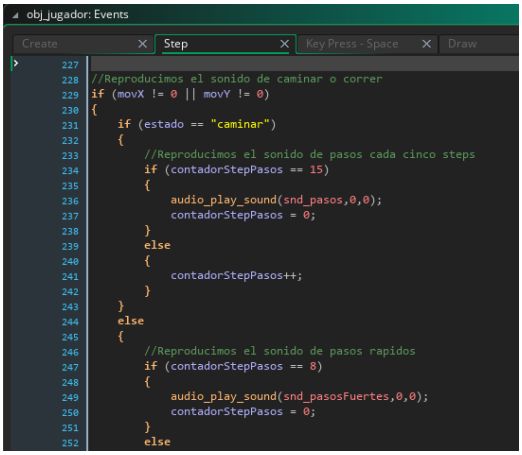


Fig. 77. Programación de la reproducción de un efecto de sonido

Una vez visto de qué forma se implementa un efecto de sonido en el juego, el resto de efectos que se contemplaron en el capítulo de diseño se van realizando de la misma forma, en cada caso identificando el suceso que ocasiona dicho efecto. Por ejemplo, el efecto de sonido de un disparo de la pistola básica se reproduce cuando se haga click izquierdo teniendo el protagonista equipado la pistola básica y con al menos una bala restante.

La implementación de las pistas de música utilizadas en las diferentes pantallas se realiza de igual forma. En este caso, la reproducción del recurso de sonido comienza

cuando se transite a cada una de las pantallas del juego y finaliza cuando se abandone dicha pantalla.

6. EVALUACIÓN DE LA SOLUCIÓN

6.1. Usuarios de evaluación

En primer lugar, se describirá de forma general el perfil de usuarios que han participado en la evaluación del producto. La idea ha sido escoger usuarios que estén incluidos en el público objetivo del videojuego, ya que pueden aportar una información bastante más fiable, a tener en cuenta para evaluar el producto, y aportar críticas constructivas sobre el mismo. Aun así, se han tenido en cuenta algunos usuarios inexpertos en videojuegos o manejo del ordenador ya que se ha considerado que podrían aportar opiniones desde un punto de vista diferente.

En total, en la fase de evaluación han participado 25 usuarios, que principalmente han sido obtenidos mediante tres vías: familiares, amigos y contactos por redes sociales, principalmente a través de una cuenta personal de la red social Twitter.

Todos ellos han sido informados del propósito académico de los experimentos, de la ausencia de fines lucrativos de dicha tarea y del tratamiento anónimo de los datos obtenidos a través de los cuestionarios y las entrevistas. Desde el punto de vista de la evaluación, la única exigencia que se les ha pedido ha sido que respondieran de forma honesta y lo más crítica posible dentro de sus conocimientos o experiencia con videojuegos.

A continuación, se describen los dos grupos de usuarios que se han podido diferenciar dentro del conjunto de usuarios que han participado de la evaluación:

6.1.1. Usuario experto en videojuegos

Este perfil de usuario incluye a personas que usan de forma cotidiana el ordenador y juegan habitualmente a videojuegos en esta plataforma. En este grupo se engloban la amplia mayoría de los usuarios que han participado en los experimentos y, debido a que todos ellos han sido obtenidos a través de amigos o contactos por redes sociales, se incluyen en una franja de edad que va desde los 18 a los 35 años aproximadamente.

De este tipo de usuarios no se esperan muchas críticas respecto a la dificultad a la hora de instalar y manejar los controles del videojuego. Sin embargo, se esperan la mayor parte de las críticas respecto a la jugabilidad, el aspecto gráfico o posibles mejoras que les habrían gustado disponer en el desarrollo de una partida.

6.1.2. Usuario inexperto en videojuegos

Este perfil de usuario incluye a personas que no usan de forma cotidiana el ordenador y no juegan habitualmente a videojuegos, ya sea en plataforma de ordenador u otras. En este grupo se engloban algunos de los usuarios participantes, obtenidos principalmente de familiares de una edad más avanzada que las pertenecientes al grupo anterior.

De este tipo de usuarios se esperan críticas respecto a la dificultad a la hora de instalar y manejar los controles del videojuego. Sin embargo, no se esperan muchas críticas respecto a la jugabilidad, el aspecto gráfico o posibles mejoras que les habrían gustado

disponer en el desarrollo de una partida, debido a la falta de conocimiento o experiencia previa con videojuegos. Este tipo de usuarios siempre serán supervisados, ya que no se espera que completen el experimento sin ayuda.

6.2. Experimentos realizados

En este apartado se detallará de qué forma se han planteado los experimentos sobre el conjunto de usuarios. Era importante mantener el mismo tipo de experimento sobre todos los usuarios que lo hiciesen de forma no supervisada, para no desvirtuar los resultados obtenidos. En este tipo de experimento se ha ofrecido una pequeña guía de cómo realizar la evaluación de forma autónoma.

De otra forma, se tiene el experimento supervisado, dónde el propio creador ha estado observando, anotando resultados y resolviendo dudas teniendo al usuario presente. Es importante decir que la mayoría de los experimentos se han realizado de forma no supervisada, por disponibilidad de tiempo y para mayor comodidad para los usuarios.

6.2.1. Experimento supervisado

Este tipo de experimento consiste en una prueba del juego bajo las siguientes condiciones:

- El juego se encuentra ya instalado en un ordenador, que dispone de teclado y ratón
- La descripción y controles de juego se facilitan al usuario
- En todo momento, el experto puede resolver cualquier duda que tenga el usuario
- El usuario debe completar, con o sin éxito, dos partidas en nivel de dificultad normal
- El experto observa cómo interactúa el usuario con el juego, qué dificultades tiene, cómo reacciona ante imprevistos y qué aspectos del juego le parecen más y menos divertidos

Una vez concluida la prueba de juego, se da un tiempo razonable al usuario para responder al cuestionario y, posteriormente, se le realiza la entrevista.

6.2.2. Experimento no supervisado

Este tipo de experimento consiste en una prueba del juego sin la participación del experto. Para ello, se ha facilitado a todos aquellos usuarios que quisiesen participar una pequeña guía para evitar cualquier tipo de duda y así lograr que el total de los experimentos se realizasen de la forma más homogénea posible.

A continuación, se presenta dicha guía, teniendo en cuenta que en el segundo punto hago una copia literal de los apartados que aparecen en el diseño de juego.

Instrucciones de evaluación

En primer lugar, muchas gracias por colaborar con la evaluación del juego como parte de mi Trabajo de Fin de Grado.
A continuación, describo cómo recomiendo realizar la evaluación:

1- Descargar e instalar el archivo "Run away or shoot! - Beta 1.exe" que se encuentra en esta misma carpeta compartida.
2- Antes de comenzar a jugar, lea la descripción del juego y controles que les detallo en este mismo documento.
3- Al menos, complete una partida en dificultad normal, ya sea con victoria o con derrota.
4- Descargue una copia del documento "Cuestionario", léalo y edítelo respondiendo a las cuestiones que en este aparecen.
5- (Opcional) Si dispone de más tiempo, descargue una copia del documento "Entrevista", léalo y edítelo respondiendo a las cuestiones que en este aparecen.
6- Una vez llegados a este punto, envíe a la dirección de correo "cristian.lopezre@gmail.com", los documentos que haya completado con el asunto "Evaluación del juego TFG".

6.3. Extracción de resultados

Una vez realizado el experimento sobre cada usuario, se presenta el cuestionario a rellenar por parte del mismo, y de forma opcional, una pequeña entrevista. Ambas técnicas tienen como objetivo obtener del usuario la experiencia de juego y la opinión sobre diferentes aspectos del producto, como pueden ser la jugabilidad, el aspecto gráfico, la dificultad en los controles de juego, extraer posibles mejoras o aspectos que han echado en falta.

6.3.1. Cuestionario

A continuación, se presenta el documento "Cuestionario" entregado a los diferentes usuarios que han participado en los experimentos realizados:

"Cuestionario de Evaluación del Juego"

Nombre y apellidos del usuario: _____ Fecha: ____/____/____

A continuación, se le presentan una serie de preguntas con el objetivo de conocer su opinión sobre diferentes aspectos del juego "Run away or shoot!".

El cuestionario se compone de un conjunto de preguntas de respuesta cerrada, en el que se indicará para cada pregunta las respuestas posibles y su significado, en caso de diferentes interpretaciones. En primer lugar, se le realizarán una serie de cuestiones para poder saber el perfil de usuario en el que encajaría y, a continuación, se le realizarán preguntas centradas en el videojuego.

6. EVALUACIÓN DE LA SOLUCIÓN

Por favor, lea detenidamente las preguntas que se realizan y contéstelas de forma honesta, ya que sus respuestas serán de gran valor. Sus respuestas se tratarán de forma confidencial y conjuntamente con el resto de usuarios de evaluación. Muchas gracias.

SECCIÓN 1: PERFIL DE USUARIO

1. ¿Con qué frecuencia utiliza el ordenador? (1- muy poco.... 5- mucho)

Respuesta:

2. ¿Se considera jugador habitual de videojuegos? (1- nunca he jugado a videojuegos... 5-juego habitualmente a videojuegos)

Respuesta:

3. ¿Juega habitualmente a videojuegos para plataforma de ordenador? (1- no soy jugador/no juego a videojuegos de ordenador... 5-juego habitualmente a videojuegos de ordenador)

Respuesta:

SECCIÓN 2: EVALUACIÓN DEL JUEGO

Tenga en cuenta que en las siguientes preguntas cuando se menciona un nivel de juego, se refiere a la dificultad normal.

4. ¿Ha tenido alguna dificultad para la instalación del juego? (1- ninguna dificultad.... 5- mucha dificultad)

Respuesta:

5. ¿De qué forma evalúa la dificultad en los controles del juego? (1- ninguna complejidad... 5- excesiva complejidad)

Respuesta:

6. ¿De qué forma evalúa la dificultad para completar con éxito un nivel de juego? (1- muy sencillo... 5-excesivamente difícil)

Respuesta:

7. ¿De qué forma evalúa la dificultad para sobrevivir, ya sea evitando o combatiendo, a los enemigos durante un nivel juego? (1- muy sencillo... 5- excesivamente difícil)

Respuesta:

8. ¿De qué forma evalúa la dificultad para descifrar las claves durante un nivel juego? (1- muy sencillo... 5- excesivamente difícil)

Respuesta:

9. Gráficamente, ¿de qué forma evalúa el juego? (1- muy bajo nivel gráfico/incoherente con la temática... 5- muy buen nivel gráfico/coherente con la temática)

Respuesta:

10. Respecto al apartado de música y de efectos de sonido, ¿de qué forma evalúa el juego? (1- muy bajo nivel / incoherente con la temática... 5- muy buen nivel / coherente con la temática)

Respuesta:

11. Indique el grado de diversión que le ha generado el juego (1- muy aburrido... 5- muy divertido)

Respuesta:

12. Sin tener en cuenta el resultado final, ¿el concepto y temática del juego le ha resultado atractivo? (1- me parece muy mala idea... 5- me parece muy buena idea)

Respuesta:

6.3.2. Entrevista

A continuación, se presenta el documento "Entrevista" entregado a los diferentes usuarios que han participado en los experimentos realizados, y que podían responder de forma opcional:

"Entrevista de Evaluación del Juego"

Nombre y apellidos del usuario: _____ Fecha: ____/____/____

A continuación, se le realizará una entrevista a un usuario de prueba del juego "Run away or shoot!" que haya aceptado libremente ser entrevistado.

La entrevista se compone de un conjunto de preguntas de respuesta abierta, con el fin de conocer la opinión del usuario respecto a diferentes aspectos del juego, dando libertad para expresar puntos favorables, desfavorables, críticas constructivas, posibles mejoras, etc.

Por favor, responda de forma honesta durante el transcurso de la entrevista. Tenga en cuenta que las respuestas se tratarán de forma confidencial y conjuntamente a las respuestas dadas por el resto de usuarios de evaluación. Muchas gracias.

1. Antes de pasar con las preguntas acerca de la experiencia de juego en sí, comente brevemente cuántas partidas ha jugado, en qué nivel de dificultad y cuántas ha conseguido completar con victoria.

Respuesta:

2. En una primera impresión, ¿te esperabas el juego así una vez leída la descripción del juego?

Respuesta:

3. Respecto a los controles del juego, ¿te han resultado complejos, o por el contrario, enseguida te has habituado a los mismos?

Respuesta:

4. ¿Qué te ha parecido la dificultad del juego para completarlo con éxito? ¿Te ha supuesto un reto completar alguna de las dificultades o, por el contrario, has conseguido completar todos los niveles sin dificultad?

Respuesta:

5. ¿Qué mecánicas o aspectos del juego te han resultado más divertidos o mejor logrados (comportamiento de los enemigos, descifrar claves, utilización de armas, humos, etc.?

Respuesta:

6. ¿Qué mecánicas o aspectos del juego te han resultado peor logrados o mejorables (comportamiento de los enemigos, descifrar claves, utilización de armas, humos, etc.)?

Respuesta:

7. ¿Has echado en falta alguna mecánica adicional, que consideras que hubiese aportado al juego?

Respuesta:

8. ¿Qué opinión tienes acerca de la duración del juego? ¿Piensas que es interesante que sea de corta duración o, por el contrario, te habría gustado que los niveles fuesen más extensos en dimensiones y duración?

Respuesta:

9. Una vez probado el juego, ¿te ha generado algún de motivación para seguir jugándolo?

Respuesta:

10. Visualmente, ¿consideras un acierto o un desacierto el tipo de gráficos utilizados en el juego?

Respuesta:

11. Respecto a la música y los efectos de sonido, ¿te han parecido coherentes con la temática del juego? ¿Echas en falta algún efecto de sonido adicional para alguna acción concreta?

Respuesta:

6.4. Análisis de resultados

A continuación, se presenta el análisis de los resultados de los cuestionarios y las entrevistas realizadas. En el caso de los cuestionarios se realiza un análisis cuantitativo de las respuestas dadas, mientras que en el caso de las entrevistas se realiza un análisis cualitativo debido a la naturaleza abierta de las respuestas solicitadas.

6.4.1. Análisis cuantitativo

En la siguiente tabla se presentan las medias de las respuestas dadas en el cuestionario para el conjunto de usuarios que han participado en la evaluación del juego:

Nº DE PREGUNTA	MEDIA OBTENIDA
1	5.00
2	4.60
3	3.94
4	1.07
5	2.18
6	3.52
7	3.30
8	3.22
9	3.73
10	3.79
11	2.96
12	4.09

Tabla 37. Resultados obtenidos de los cuestionarios de evaluación

En primer lugar, analizamos los resultados obtenidos del primer bloque de cuestiones, centrados en la confirmación de que el perfil de usuario participante en la evaluación es efectivamente el buscado: usuarios que utilizan frecuentemente el ordenador, y que en general, juegan habitualmente a videojuegos para esta plataforma.

Respecto a las cuestiones centradas en evaluar la complejidad de diferentes elementos del juego (preguntas 4 a 8), como son la instalación, los controles, las propias mecánicas del juego, los resultados obtenidos son bastante positivos. La dificultad a la hora de instalar y controlar el juego se evalúa como ninguna complejidad o escasa complejidad. Sin embargo, y como era de esperar, la complejidad de las mecánicas del juego es evaluada con una mayor puntuación. Ambos aspectos, tanto el de la supervivencia como el del descifrado de claves tienen una puntuación prácticamente idéntica, y parece equilibrada la dificultad del juego en general, teniendo una puntuación de 3.52 sobre 5.

En cuanto a la evaluación de los apartados gráficos, música y efectos de sonido del juego, se tiene una valoración bastante positiva, que roza los 4 puntos en todos ellos. Por tanto, se puede concluir que los usuarios están satisfechos en general con esta parte del juego.

Finalmente, la valoración que haces los usuarios de su grado de diversión tras haber jugado varias partidas es prácticamente de 3 puntos. Sin embargo, respecto a la idea de juego como tal les ha parecido bastante buena, ya que supera ligeramente los 4 puntos. De estos resultados se puede concluir que los usuarios piensan que existe un margen de mejora respecto a la implementación final del juego respecto a la idea diseñada.

6.4.2. Análisis cualitativo

Debido a la naturaleza abierta de muchas de las respuestas solicitadas en las entrevistas se ha optado por obtener la opinión o crítica más repetida de entre los usuarios que quisieron colaborar en estas entrevistas. Pasamos a analizar cada una de las preguntas y las respuestas obtenidas:

1. Todos los usuarios al menos han jugado 4 partidas, de las cuales se detecta que una vez completada con éxito una dificultad, pasaban a intentar de una dificultad superior. El ratio de victorias respecto a los intentos ronda entre el 25% y el 35% en la mayoría de los casos. Quizás se esperaba un ratio de victorias algo superior en la dificultad más fácil, pero en general, es lo esperado, ya que se busca que el juego sea un verdadero reto para el jugador.

2. La amplia mayoría de los usuarios se esperaban el juego **así** una vez leída la descripción del mismo. Sólo uno de los usuarios lo esperaba algo más oscuro en la parte gráfica. El feedback obtenido de esta pregunta es muy importante, ya que la coherencia del juego entre las fases de diseño e implementación es algo imprescindible a la hora de evaluarlo.

3. El aspecto de los controles de juego ha tenido respuestas de lo más variadas. Algunos de los usuarios, quizás más experimentados en juegos de ordenador o en dos dimensiones, lo encontraban cómodos y fluidos. Otros usuarios, sin embargo, encontraban los controles algo incómodos, sobre todo la tecla de acceso al inventario debido a la lejanía respecto al resto de controles.

4. Respecto a la dificultad general del juego, las respuestas obtenidas han sido coherentes con el ratio de victorias mencionado en el primer punto ya que a la gran mayoría le ha parecido lo suficientemente complejo como para plantear el juego como un reto.

5. Los aspectos mejor valorados del juego han sido, en general, el comportamiento inteligente de los enemigos y la variedad de armas que se puedan utilizar en el juego. También se ha valorado de forma positiva que no todas las armas fuesen de fuego, sino también para otro tipo de uso, como la pistola táser o las bombas de humo.

6. Los aspectos peor valorados del juego han sido la poca variedad de tipo de puzles y que los niveles no fuesen generados de forma más aleatoria. El feedback obtenido de este punto será tenido en cuenta para comentar las mejoras o proyectos futuros en este mismo trabajo.

7. Las mecánicas que han echado en falta los usuarios y les gustaría haber podido disfrutar en el juego son la inclusión de un mini mapa, poder cerrar las puertas, aleatoriedad en la generación de los niveles, o diferentes tipos de enemigos. Al igual que en el punto anterior, todas estas ideas serán tenidas en cuenta como mejoras o proyectos futuros.

8. En este punto, ha existo unanimidad en todas las respuestas. La duración para todos los usuarios ha resultado ser ideal y equilibrada respecto a la dificultad buscada en el juego. Era un aspecto que preocupaba en la fase de diseño del juego, ya que el equilibrio exacto entre la duración del juego y la dificultad era difícil de estimar en primeras fases del proyecto.

9. Respecto a la motivación para seguir jugando una vez completadas algunas partidas, las respuestas han variados en ambos sentidos. Algunos usuarios

plantean que tendrían motivación en caso de disponer de más niveles, si además estos fuesen generados de forma aleatoria. Otros usuarios indican que jugarían simplemente de forma casual para echarse unos ratos divertidos cuando no dispusiesen de mucho tiempo para el ocio.

10. En general, la parte gráfica del juego está bastante bien valorada, salvo algún usuario que indica que los gráficos retro no son muy de su gusto. Este punto era bastante crítico, ya que la parte gráfica es una de las que más esfuerzos en tiempo ha llevado en la fase de implementación del juego, y que la respuesta haya sido en general positiva recompensa el tiempo empleado.

11. Finalmente, la música y los efectos de sonido empleados en el juego también han tenido una respuesta positiva. Excepto un usuario que se quejó de la música utilizada en la pantalla de título, todos los demás valoraron como coherente la música empleada en las diferentes pantallas y no echaron en falta más efectos de sonido para las acciones producidas durante el transcurso de una partida.

Una vez analizados cada uno de los puntos tratados en las entrevistas, se puede concluir como positiva la valoración general del juego, teniendo en cuenta aspectos de jugabilidad, mecánicas de juego, y parte estética. Las críticas e ideas de mejoras del juego comentadas por los usuarios son recibidas con buen agrado y serán tenidas en cuenta en el capítulo dedicado a ese propósito en esta misma memoria.

6. EVALUACIÓN DE LA SOLUCIÓN

7. METODOLOGÍA Y PLANIFICACIÓN DEL PROYECTO

7.1. Metodología de trabajo

El trabajo de fin de grado descrito en esta memoria queda enmarcado dentro de un proceso de ingeniería del software con una serie de fases de desarrollo claramente diferenciables.

En primer lugar, una fase de análisis cuyo objetivo es la identificación del problema y su solución, para la que se extrae un conjunto de requisitos de capacidad y de restricción. En esta fase también se pueden incluir como tareas el estudio de la competencia y sustitutivos, así como el estudio del estado del arte de las diferentes herramientas a utilizar para el desarrollo del software y de su contexto.

La siguiente fase es la fase de diseño, cuyo objetivo es, partiendo de los objetos resultantes de la fase de análisis, detallar las características y requisitos de la solución propuesta. Los objetos resultantes de esta fase de diseño son los prototipos de las diferentes pantallas que componen el videojuego y la descripción detallada de las mecánicas de juego que aparecen en el mismo.

A partir de los objetos resultantes del diseño se pasa a la fase de implementación, cuyo objetivo es el de codificar la solución. Para ello, se debe seguir de forma lo más estricta posible todas las características definidas en la lista de requisitos y, posteriormente, detalladas en la fase de diseño.

Finalmente, se tiene la fase de evaluación, cuyo objetivo principal es el de comprobar el grado de satisfacción de los usuarios respecto al producto final. Hay que tener en cuenta que la mayoría de errores o bugs son detectados durante la fase de implementación, ya que no se pretende que los usuarios de evaluación tengan que acarrear con esta tarea.

Es importante añadir que, en todas las fases de desarrollo, antes de pasar a la siguiente, se han realizado tareas de revisión exhaustivas para verificar que todas las características derivadas de la fase tienen su proyección en la fase de desarrollo actual. En caso negativo se permite regresar hacia una fase anterior para realizar modificaciones, generalmente no demasiado críticas en el software a desarrollar, y así cumplir con el objetivo de la aplicación de la metodología de desarrollo.

Concluyendo con lo expuesto anteriormente se puede afirmar que la metodología de desarrollo utilizada durante la realización de este trabajo ha sido la de un desarrollo en cascada con retroalimentación, como se muestra en la siguiente figura:

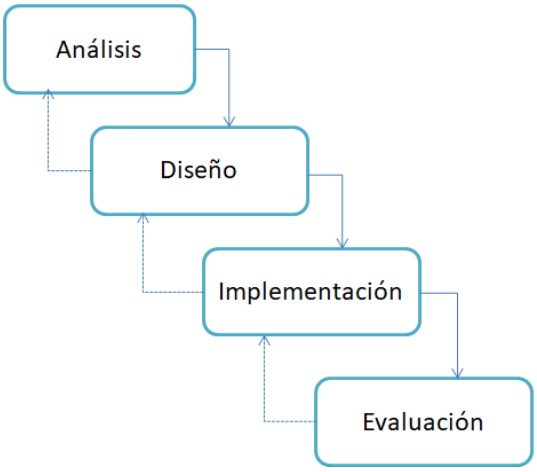
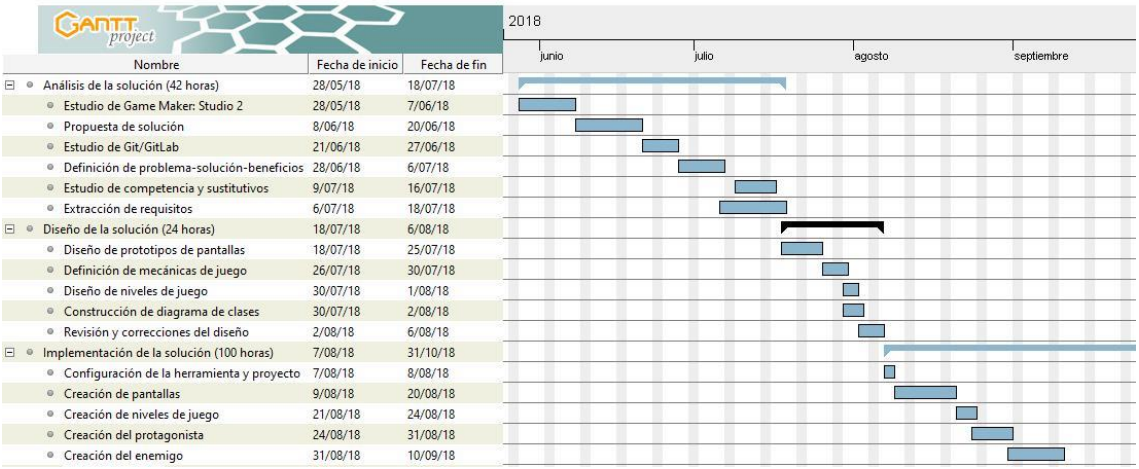


Fig. 78. Metodología de desarrollo en cascada con retroalimentación

7.2. Planificación

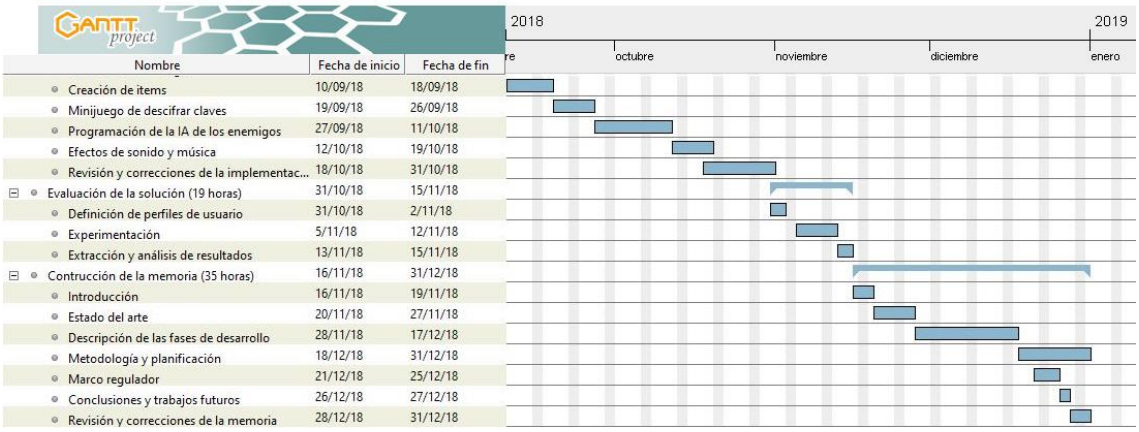
Antes de comenzar la realización del trabajo se estimó un total de tiempo estimado de 180 horas de duración del trabajo, teniendo en cuenta proyectos similares de acuerdo al contexto de un trabajo de fin de grado. Se planteó una división del tiempo por fases de alrededor del 40% para la fase de implementación, un 20% para la fase de análisis, un 20% para la fase de diseño, un 10% para la fase de evaluación y un 10% para la construcción final de la memoria, teniendo en cuenta que durante todas las fases se iría redactando la documentación correspondiente.

A continuación, se muestra el diagrama Gantt resultante del tiempo dedicado a cada una de las tareas y fases del desarrollo de este trabajo. Debido al tamaño que tenía dicho diagrama se ha optado por separar en varias figuras la totalidad del mismo para una mejor visualización:



Nombre	Fecha de inicio	Fecha de fin
Análisis de la solución (42 horas)	28/05/18	18/07/18
Estudio de Game Maker: Studio 2	28/05/18	7/06/18
Propuesta de solución	8/06/18	20/06/18
Estudio de Git/GitLab	21/06/18	27/06/18
Definición de problema-solución-beneficios	28/06/18	6/07/18
Estudio de competencia y sustitutivos	9/07/18	16/07/18
Extracción de requisitos	6/07/18	18/07/18
Diseño de la solución (24 horas)	18/07/18	6/08/18
Diseño de prototipos de pantallas	18/07/18	25/07/18
Definición de mecánicas de juego	26/07/18	30/07/18
Diseño de niveles de juego	30/07/18	1/08/18
Construcción de diagrama de clases	30/07/18	2/08/18
Revisión y correcciones del diseño	2/08/18	6/08/18
Implementación de la solución (100 horas)	7/08/18	31/10/18
Configuración de la herramienta y proyecto	7/08/18	8/08/18
Creación de pantallas	9/08/18	20/08/18
Creación de niveles de juego	21/08/18	24/08/18
Creación del protagonista	24/08/18	31/08/18
Creación del enemigo	31/08/18	10/09/18

Fig. 79. Diagrama de Gantt (parte 1)



Nombre	Fecha de inicio	Fecha de fin
Creación de items	10/09/18	18/09/18
Minijuego de descifrar claves	19/09/18	26/09/18
Programación de la IA de los enemigos	27/09/18	11/10/18
Efectos de sonido y música	12/10/18	19/10/18
Revisión y correcciones de la implementac...	18/10/18	31/10/18
Evaluación de la solución (19 horas)	31/10/18	15/11/18
Definición de perfiles de usuario	31/10/18	2/11/18
Experimentación	5/11/18	12/11/18
Extracción y análisis de resultados	13/11/18	15/11/18
Contrucción de la memoria (35 horas)	16/11/18	31/12/18
Introducción	16/11/18	19/11/18
Estado del arte	20/11/18	27/11/18
Descripción de las fases de desarrollo	28/11/18	17/12/18
Metodología y planificación	18/12/18	31/12/18
Marco regulador	21/12/18	25/12/18
Conclusiones y trabajos futuros	26/12/18	27/12/18
Revisión y correcciones de la memoria	28/12/18	31/12/18

Fig. 80. Diagrama de Gantt (parte 2)

Una vez concluido el proyecto, y teniendo en cuenta que el total de horas empleadas ha sido de 220 horas, se pasa a comparar la estimación inicial respecto al tiempo final empleado en cada una de las fases:

- En la fase de análisis se han dedicado un total de 42 horas, que suponen un 19% del total dedicado al proyecto. Prácticamente es idéntica al tiempo estimado inicialmente, esto se debe a que durante esta fase no han ocurrido prácticamente imprevistos y las tareas planteadas se cumplieron en los tiempos establecidos.

- En la fase de diseño se han dedicado un total de 24 horas, que suponen un 11% del total dedicado al proyecto. En este caso, el tiempo estimado era bastante superior al finalmente dedicado. Probablemente, se puede deber a que durante la estimación del tiempo dedicado a esta fase no se tenía muy clara la estructura de tareas que se iban a incluir, debido a la naturaleza del software a desarrollar y la poca experiencia con el mismo.

- En la fase de implementación se han dedicado un total de 100 horas, que suponen un 45% del total dedicado al proyecto. Es ligeramente superior al tiempo estimado inicialmente, y esto puede deberse a las dificultades encontradas durante la implementación de la parte gráfica del videojuego, en el diseño de los personajes, que claramente es lo que más tiempo de desarrollo consumió.

- En la fase de evaluación se han dedicado un total de 19 horas, que suponen un 9% del total dedicado al proyecto. La estimación prácticamente se ha cumplido en esta fase, debido a que las tareas planteadas se han cumplido en los tiempos establecidos sin imprevistos.

- Para la construcción de la memoria se han empleado un total de 35 horas, que suponen un 16% del total dedicado al proyecto. En este caso, el tiempo estimado era algo inferior, y esto se ha debido principalmente a que durante la fase de implementación no se documentó lo suficiente a medida que se iba codificando el videojuego y se ha tenido que dedicar un tiempo adicional al previsto.

7.3. Presupuesto

Una vez calculadas las horas de trabajo dedicadas a la elaboración del trabajo en su totalidad, resultando en un total de 220 horas, se pasa a detallar el presupuesto detallado y dividido por costes de personal, costes de hardware y costes de software.

Para realizar el cálculo del coste de personal se ha tomado como perfiles de trabajadores el de un ingeniero informático / analista para el desarrollador del trabajo y el sueldo de un consultor informático especializado en el área de videojuegos para la labor de asesoramiento realizada por el tutor. Se ha tomado un sueldo bruto promedio de 40€/h para el primero y de 80€/h para el segundo.

Para realizar el cálculo de los costes de hardware y software se han tenido en cuenta los periodos de amortización de los mismos y el tiempo de utilización para la elaboración del proyecto, de forma que el coste de cada elemento sea la parte proporcional y no el total de la adquisición del producto o la licencia en cuestión.

COSTES DE PERSONAL			
Nombre	**Coste por hora**	**Horas trabajadas**	**Coste total**
Cristian López Reviriego	40 €/h	220 h	8.800,00 €
Jorge Ruiz Magaña	80 €/h	30 h	2.400,00 €
		Total:	

COSTES DE HARDWARE				
Descripción	**Coste**	**Uso**	**Amortización**	**Coste imputable**
PC sobremesa (Intel Core i7-930 2,8 GHz)	1100,00 €	7 meses	48 meses	160,41 €
			Total:	160,41 €

COSTES DE SOFTWARE				
Descripción	**Coste**	**Uso**	**Amortización**	**Coste imputable**
Microsoft Windows 10	145,00 €	7 meses	48 meses	21,15 €
Microsoft Office 2016	150,00 €	7 meses	48 meses	21,87 €
Game Maker Studio 2 Creator Edition	39,00 €	5 meses	12 meses	16,25 €
Balsamiq Mockups 3 for Desktop	89,00 €	6 meses	12 meses	44,5 €
Adobe Photoshop	140,00 €	5 meses	12 meses	58,33 €
			Total:	162,10 €

Tabla 38. Presupuesto desglosado

Una vez calculados los costes totales desglosados por categorías, se pasa a calcular el coste total del proyecto:

COSTE TOTAL	
Concepto	**Coste**
Personal	11.200,00 €
Hardware	160,41 €
Software	162,10 €
Total sin impuestos	11.522,51 €
IVA (21 %)	2.419,73 €
Total:	**13.942,24 €**

Tabla 39. Presupuesto total

En caso de querer aplicar un porcentaje de beneficios e imprevistos respecto al total presupuestado, habría que calcularlos previamente a la aplicación del IVA. Por ejemplo, si quisiéramos calcular el coste del proyecto aplicando un porcentaje del 25% de beneficios y un 10% de costes imprevistos, el total sin impuestos alcanzaría la cifra de 15.555,39 €. Aplicando posteriormente el 21% de IVA el proyecto tendría un presupuesto total de 18.822,02 €.

7. METODOLOGÍA Y PLANIFICACIÓN DEL PROYECTO

8. MARCO REGULADOR

8.1. Propiedad intelectual: estado de la cuestión, problemas y medidas de protección

Se sabe que la legislación de las diferentes materias que regulan nuestra existencia nunca ha evolucionado al mismo ritmo con el que lo hace la realidad social, económica o tecnológica. Esta diferencia de velocidades es, en ocasiones, mayor cuando hablamos de nuevos productos o servicios relacionados con las nuevas tecnologías, creando una gran necesidad de un marco regulador.

En relación al plano de la Propiedad Intelectual en áreas como el desarrollo de software, o, como en el caso de este trabajo, de los videojuegos, sí que se ha experimentado una notable evolución en esa actualización o revisión de las leyes. En concreto, esta revisión ha tenido su reflejo en la Directiva 2001/29/CE y en la Leyes 23/2006 y 21/2014 de reforma de la Ley de Propiedad Intelectual (en adelante LPI).

Durante los últimos años la LPI se ha encargado de recoger una serie de modificaciones encaminadas a preservar los derechos patrimoniales del titular de un programa informático o de un videojuego. La primera referencia a la protección de un programa de ordenador en la LPI, tal como recuerda Corripio Gil Delgado [32], se contempló en el antiguo artículo 8 de la Ley 16/1993, actualmente derogado, que incluyó "como un nuevo acto de infracción de los derechos de autor la puesta en circulación o tenencia con fines comerciales de cualquier instrumento cuyo único uso fuera facilitar la supresión o neutralización no autorizadas de cualquier dispositivo técnico utilizado para proteger un programa de ordenador."

Años más tarde, la Directiva 2001/29/CE obligó a los Estados Miembros a proteger las medidas tecnológicas para las obras comercializadas en soporte digital. Esta regulación es más moderna y extensa que la otorgada al software en la antigua LPI.

Desde aquella Directiva y la anterior LPI, el marco regulador vigente ha progresado teniendo en cuenta los avances tecnológicos y las diferentes problemáticas que iban surgiendo respecto a la materia. Actualmente, este nuevo marco se regula en el Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, y que además se ha ido actualizado en los sucesivos años. En concreto, la regulación de esta materia se encuentra en el Libro Primero, Título VII Programas de ordenador, artículos 95 a 104 de dicha Ley.

Sin embargo, y debido a que no existe un tratamiento específico como tal para el caso de un producto como el de un videojuego, se debe acudir a las normas y que en cierta forma son asimilables con las adaptaciones que se requieren para este tipo de productos, debido a la similitud existente entre un programa de ordenador y un videojuego, dentro del plano legal.

Para la protección de los Derechos de Propiedad Intelectual (en adelante DPI) de un producto como el de los videojuegos, los principales despachos de abogados recomiendan registrar las diferentes partes de las que se compone. Las partes en las que

se suele disgregar un videojuego con la finalidad de la protección de los DPI son el código fuente, la parte audiovisual, y el guion o historia. En el caso del código fuente y los aspectos del guion o la historia se deben tratar como el registro de una obra literaria, mientras que el caso de la parte audiovisual del videojuego se debe tratar como una obra audiovisual.

Los programas de ordenador, al igual que los videojuegos, aunque puede ser considerados nuevos tipos de obras diferentes a los de una obra literaria, no son tratados por los legisladores propiamente como nuevas categorías de creaciones, sino que son asimilado como creaciones literarias, como se dice en el artículo 10.1 de la LPI. Esta inclusión dentro de la categoría de una obra literaria para la aplicación de la LPI ya la menciona De Miguel Asensio [33] "determinante de la protección de los programas de ordenador mediante el derecho de autor en virtud de su asimilación a las obras literarias es la circunstancia de que todo programa se expresa mediante ciertos tipos de lenguaje en lo que se conoce como el código fuente y el código objeto del programa".

Se hace necesario también definir, desde el punto de vista de lo que entiende la LPI, dos conceptos muy relevantes como son el de programa de ordenador y autor de un programa y que explicamos a continuación:

- Programa de ordenador: toda secuencia de instrucciones o indicaciones destinadas a ser utilizadas, directa o indirectamente, en un sistema informático para realizar una función o para obtener un resultado determinado, sin tener en cuenta su forma de expresión. Dentro de un programa de ordenador, además, hay que incluir la documentación o manuales de uso del programa, por lo que gozarán de los mismos derechos de protección. Según la LPI, el programa será protegido únicamente si fuese original, en el sentido de ser una creación intelectual propia de su autor. Asimismo, las sucesivas versiones del programa dispondrán de los mismos derechos de protección que la primigenia.

- Autor de un programa: persona o grupo de personas físicas que hayan creado el programa, o la persona jurídica que sea propietaria de los derechos de autor. Esto quiere decir que a pesar de que un trabajador de una compañía se pueda considerar como parte coautora de un determinado programa, con fines legales los derechos de autoría le corresponden al empresario.

Un aspecto bastante interesante es el de la duración de los DPI de un programa. En el caso de que el autor sea una persona individual los derechos se mantienen durante la vida del autor y hasta setenta años tras su muerte. En el caso de que el autor sea un grupo de personas, los derechos se mantienen hasta que transcurran setenta años después de la muerte del último de los coautores supervivientes. En el caso de que el autor sea una persona jurídica, los derechos se preservan durante setenta años computándose a partir del 1 de enero del siguiente año al de la publicación del programa.

Respecto a la parte audiovisual de un videojuego, se recomienda proteger de la misma forma que una obra audiovisual, de acuerdo a lo que expuesto en los artículos 86 a 94 de la LPI. Para la LPI, una obra audiovisual es "una creación expresada mediante una serie de imágenes asociadas, con o sin sonorización incorporada que estén destinadas esencialmente a ser mostradas a través de aparatos de proyección o por cualquier otro

medio de comunicación pública de la imagen y del sonido, con independencia de la naturaleza de los soportes materiales de dichas obras".

En algunas ocasiones, los videojuegos son adaptaciones de obras cinematográficas o de otro tipo de creaciones, para lo que la LPI recoge que para este tipo de transformaciones de una obra en otra se presume que el autor o autores de la obra predecesora cede los derechos de explotación sobre la obra derivada.

Como ya se ha mencionado, una de las medidas de protección de los DPI del videojuego sería el registro de cada una de las partes. Sin embargo, es importante aclarar que el hecho de registrar una obra no tiene carácter constitutivo de la autoría, es decir, no es necesario registrar una obra para considerar que una obra es de quien la hizo. Por lo tanto, el registro se trata más bien de una medida de protección frente a terceros.

La LPI atribuye una serie de derechos a los autores. Puesto que esos DPI son fácilmente utilizables por terceros sin el consentimiento del autor o titular, mediante el Registro de la Propiedad Intelectual se dota de un mecanismo adicional de protección que opera en el ámbito de la prueba, consiguiendo, de esta manera, una mayor seguridad jurídica en el tráfico de los derechos.

Una vez inscrita una obra en el Registro, los datos que en él figuran gozan de la condición de prueba cualificada, es decir, que ante una eventual discrepancia en cuanto a la autoría o titularidad de derechos sobre una obra, se presume que los datos del Registro son exactos, y judicialmente se resuelve en favor del autor que figura en el mismo.

Otra medida de protección sería acudir a la vía penal, una vez estos DPI han sido vulnerados. Los delitos contra la Propiedad Intelectual se ubican en la Sección 1ª del Capítulo XI del Título XIII del Código Penal, que incluye los artículos 270 a 272, en el que se indican penas que van desde multas económicas hasta la pérdida de libertad para los infractores.

Para la aplicación de este tipo de penas, se deben cumplir una serie de características que se pueden resumir en que el que lleva a cabo la violación de los derechos lo haga con un ánimo de lucro, que dicha actuación se haga sin la autorización del titular de los derechos y que la actuación del sujeto infractor sea lo suficientemente idónea para causar un perjuicio a un tercero que en la mayoría de casos son el titular de los derechos.

En función del tipo de personas que vulneren estos DPI, las penas variarán. En el caso de que el infractor sea una persona física, y para casos de comercialización o distribución ilegal de un producto, pueden llegar hasta los seis años de prisión. En el caso de que el infractor sea una persona jurídica, las multas económicas pueden llegar a ser el cuádruple del beneficio obtenido con la práctica de dicha actividad ilegal.

8.2. Protección de datos de carácter personal

En este apartado, dada la naturaleza de este trabajo, se realizará una breve exposición de las principales exigencias con respecto al tratamiento de la información de terceros y ejemplificarlo con algún caso reciente en torno a los videojuegos. Por otro lado, se hará una breve mención a las principales novedades, dada la reciente aprobación de una nueva ley de protección de datos, que podrían afectar al producto desarrollado en este trabajo.

Todas las empresas, sociedades, autónomos, comunidades, asociaciones y administraciones públicas de los Estados miembros de la UE, que manejen mínimamente información personal de personas físicas están obligados a seguir los protocolos de tratamiento de dicha información contemplada en la Ley estableciendo la creación de ficheros con dichos datos, que deberán custodiar. Este deber de custodia, es proporcional a la sensibilidad de la información personal que se requiera. Por ejemplo, no se establecerán los mismos protocolos de seguridad sobre aquellos datos como el nombre identificador de un jugador, que sobre los datos como el DNI o datos de cuenta bancaria. Todo esto sin olvidar que, aunque los datos no sean sensibles, se crea una obligación del custodio y, en consecuencia, una responsabilidad en caso de filtraciones o tratamiento inadecuado de los datos. De igual forma, la Ley de protección de datos, obliga a los custodios de dichos datos, permitir la accesibilidad a dichos datos por parte de las personas titulares de los mismos para poder ejercitar todos aquellos derechos con respecto a su información que la ley les atribuye, entre los que se encuentran, los derechos de rectificación, actualización o eliminación de sus datos.

Es importante y aconsejable, en el plano del trabajo aquí desarrollado, minimizar en la medida de lo posible todos los datos personales solicitados respecto a los usuarios del videojuego, ya que como se ha visto anteriormente implica una mayor responsabilidad y un mayor coste de mantenimiento y seguridad de los mismos. El objetivo último en este aspecto es el de evitar cualquier tipo de problema legal, como los que se han conocido últimamente como es del caso expuesto a continuación.

En el caso del videojuego *Pokemon Go*, que como nos recuerda Biurrun Abad [34] “el principal problema (al usar la aplicación), que parece ser que va siendo poco a poco solucionado con las actualizaciones que Niantic va lanzando, es la geolocalización. Como nos detalla Guillermo Cernuda, “la aplicación recoge la geolocalización exacta y detallada del usuario, y mientras juega y camina por la ciudad, envía información sobre su ubicación, lo que permite a sus creadores poseer un mapa exacto sobre sus hábitos y movimientos. Hasta hace poco, a dicha información se sumaba la posibilidad de acceso por los desarrolladores a toda la cuenta Google de los usuarios. Datos que, según las condiciones generales de la app, podían compartirse con terceros para llevar a cabo “investigaciones y análisis, perfiles demográficos y otros fines similares”, o con los gobiernos o agentes de la ley, para frenar “actividades ilegales o inmorales”. No obstante, tras la polémica sobre el completo acceso a la cuenta Google, las condiciones del juego han cambiado, recogiéndose tan solo el nombre de usuario y la dirección de correo”.”

Este ejemplo es una muestra clara de una utilización malintencionada de los datos recogidos de los jugadores en su tiempo de juego. Desde el punto de vista del jugador hay que tener especial cuidado con los permisos que solicitan este tipo de aplicaciones y

los datos que recogen de nuestros dispositivos, como es el caso del acceso al GPS y si está información va a ser almacenada con algún otro propósito.

Respecto a la nueva Ley de protección de datos que opera en nuestro país se trata de una evolución de la anterior LOPD, que ahora se conoce como Ley Orgánica de protección de Datos y Garantía de los Derechos Digitales (en adelante LOPDGDD). Debido a su extensión, solo se mencionarán aquellos aspectos más relevantes para el presente trabajo.

Ya en su preámbulo, nos recalca el enfoque que se le ha querido dar a esta nueva regulación, "La protección de las personas físicas en relación con el tratamiento de datos personales es un derecho fundamental protegido por la Constitución española.". De igual forma, el nombre actualizado de la Ley nos da pistas sobre las principales modificaciones que se han implementado en la regulación, y es que se el hecho de que se denomine "Garantía de los Derechos Digitales" implica una amplia modificación que hace hincapié en estos nuevos Derechos y que encaja perfectamente con el ámbito de este trabajo.

La nueva realidad de los videojuegos es que la mayoría de ellos se ejecutan en forma online. Así pues, la nueva Ley establece una serie de nuevos protocolos, con respecto a los derechos de los usuarios y sus datos. Estos protocolos están enfocados en la facilitación o accesibilidad de los medios, por medio de los cuales los usuarios puedan, acceder a sus datos y poder ejercitar todos los derechos que establece la Ley, como, por ejemplo, los de rectificación, actualización o eliminación. Además, se regula el modo en que debe informarse a las personas acerca del tratamiento de sus datos optándose, específicamente en el ámbito de internet, por un sistema de información por capas que permita al ciudadano conocer de forma clara y sencilla los aspectos más importantes del tratamiento, pudiendo acceder a los restantes a través de un enlace directo.

En el caso de que los jugadores sean menores de edad, la Ley establece que, a partir de los 14 años, los menores podrán prestar consentimiento de manera autónoma. También se regula el derecho a solicitar la supresión de los datos facilitados por el propio menor o por terceros durante su minoría de edad.

Por otro lado, también hay que recalcar que el videojuego desarrollado en este TFG está dentro de los supuestos establecidos en la Ley de servicios de la sociedad de la información. Y, por tanto, le afecta otra de las novedades incluidas en la LOPDGDD, que no es otra que la regulación de lo que habitualmente se conoce como "derecho al olvido", que otorga al usuario el poder de decisión sobre la imposibilidad de difusión de sus datos personales almacenados en una aplicación concreta por parte de los propietarios de dicha aplicación.

8.3. Explotación económica del videojuego

Como se comentó al principio de este trabajo, el sector del videojuego se ha posicionado como uno de los sectores con mayor crecimiento en el mercado español y global. Esta evolución ha motivado la creación de estructuras jurídicas cada vez más complejas para su explotación, dada la intervención de más actores e intereses.

A continuación, se van a plantear las estructuras jurídicas más interesantes para la posible explotación económica del videojuego desarrollado en este trabajo:

- Sociedad mercantil

Esta estructura sería la más adecuada en el caso de querer realizar una explotación económica del videojuego por cuenta propia. La constitución de una sociedad mercantil para la explotación del videojuego no tiene que ser estrictamente posterior al desarrollo del videojuego, sino que podrá ser constituida en paralelo a su desarrollo. En general, y teniendo en cuenta el tipo de producto del que se está hablando, se suelen tratar de sociedades formadas por más de un individuo, por lo que se hace necesario establecer lo que se conoce como un acuerdo societario. Este acuerdo sirve para delimitar las implicaciones societarias de dicha colectividad, más allá de los típicos acuerdos que los estatutos de la sociedad y que la ley exige. Otro tipo de acuerdo a establecer dentro de una sociedad de este tipo, y cuando se plantea la inclusión de terceros inversores, es lo que se conoce como pactos parasociales.

- Venta del videojuego a terceros

En este caso la estructura jurídica se simplifica, ya que se realiza como una venta de los derechos de explotación del producto desarrollado. Un caso muy interesante es de la venta de alguna de las partes del videojuego, y no en su totalidad. Por ejemplo, podría darse la situación de que la idea del videojuego fuese interesante para un comprador, pero no la implementación realizada, por lo que el objetivo de este es únicamente comprar la idea y desarrollar un producto propio con sus medios. En el caso del videojuego creado con este trabajo este planteamiento encaja perfectamente, debido a las limitaciones encontradas a la hora de desarrollar, por ejemplo, la parte audiovisual del mismo. Sin embargo, alguna empresa desarrolladora con un mayor potencial podría encontrar interesante la idea del juego y sus mecánicas.

Por último, añadir que hay un aspecto con el que tener especial cuidado cuando se venden los derechos de explotación de nuestro producto a otras empresas o personas. Pese a que esta venta de derechos se puede hacer tanto en una única cantidad fija como en porcentajes sobre el beneficio que extraiga ese inversor de los derechos de explotación, se aconseja fijar de la forma más amplia posible los términos del acuerdo. Se han dado casos, algunos muy recientes, como por ejemplo el del escritor polaco Andrzej Sapkowski. Siendo el creador de la saga del brujo Geralt de Rivia, en un principio vendió los derechos de explotación a CD Project, una empresa de desarrollo de videojuegos del mismo país, por una cifra inferior a los diez mil euros. Obviamente, no esperaba un gran éxito en el videojuego que podrían desarrollar en dicha compañía, y sucedió todo lo contrario, convirtiéndose en una de las sagas de juegos más vendidos de estos últimos años.

9. CONCLUSIONES

Una vez desarrollado este trabajo fin de grado es de especial interés redactar unas conclusiones acerca de los objetivos fijados en unas primeras fases del trabajo y de qué forma se reflejan en los resultados finalmente obtenidos con el desarrollo de la solución planteada.

Para una mayor claridad se optará por enumerar cada uno de los objetivos planteados al principio del trabajo y justificar su cumplimiento.

1- Ampliar la oferta de juegos casual existentes en el mercado

El videojuego ha sido implementado de forma satisfactoria cumpliendo la totalidad de los requisitos planteados en la fase de análisis. En cuanto al aspecto casual del juego, se refleja perfectamente en varios de los requisitos del videojuego implementado. Por ejemplo, los tiempos de duración de una partida no superan los 12 minutos, el juego dispone de varios niveles de dificultad, no posee una curva de aprendizaje excesivamente elevada, dispone de interfaces bastante intuitivos, y los gráficos del juego van en consonancia con las tendencias habituales en los juegos casuales más exitosos en la actualidad, como son los de gráficos tipo retro o pixel art.

2- Crear un juego de fácil portabilidad

Si bien es cierto que la licencia adquirida solo permitía desplegar el juego en plataforma de PC, durante el diseño y la implementación del juego se ha tenido presente esta característica, no complicando en exceso aspectos a tener en cuenta en la portabilidad, como los controles del juego, de forma que sea sencillo cambiar los eventos que controlan al personaje o los diferentes menús de las pantallas de juego.

3- Crear un juego fácilmente escalable

Al utilizar un paradigma de programación orientado a objetos es bastante intuitivo observar de qué forma se han implementado los elementos de juego comentados anteriormente y utilizarlos como plantillas para la creación de otros elementos personalizados. Por tanto, el código resultante es de fácil lectura y comprensión para futuras mejoras del videojuego, en aspectos como la ampliación de niveles de juego, tipos de enemigos y objetos.

4- Aplicación de una metodología de desarrollo software tradicional

Como se ha descrito en esta memoria, se ha utilizado una metodología de desarrollo en cascada con retroalimentación, cumpliendo cada una de las fases de desarrollo de acuerdo con las tareas típicas de cualquier otro tipo de software. Las principales diferencias existentes entre las distintas actividades típicas en las metodologías de desarrollo para un software tradicional y un videojuego como el desarrollado han tenido su localización en la fase de diseño, donde quizás se le ha dado un mayor protagonismo a actividades como el diseño de prototipos y la definición de mecánicas de juego, propias de este tipo de software.

9. CONCLUSIONES

Una vez analizados cada uno de los objetivos planteados, se puede concluir que el trabajo realizado cumple con los mismos y resuelve de forma exitosa el problema planteado al inicio del trabajo.

Respecto a las principales dificultades encontradas durante la realización de este trabajo se pueden mencionar las siguientes:

- Durante la fase previa al análisis de la solución, de pura creatividad, hubo que realizar una serie de iteraciones hasta dar con una propuesta de solución que se pensase abarcable en unos plazos razonables dentro del contexto de un trabajo de fin de grado y que, a su vez, pudiese dar una respuesta exitosa a los objetivos planteados al inicio del trabajo.

- Durante la transición de la fase de análisis a la fase de diseño hubo que realizar algunos ajustes a los requisitos planteados en un primer momento, debido a que no quedaban correctamente reflejados en la siguiente fase de desarrollo.

- Ya en la implementación de la solución, una de las principales dificultades encontradas estuvo en la implementación de la parte gráfica del videojuego, debido principalmente a mi propia inexperiencia con el uso de herramientas gráficas a un nivel más avanzado. Otro de los elementos más complejos de implementar, pero que a la vez suponía un desafío muy gratificante de superar, fue la creación de la inteligencia artificial de los personajes controlados por la máquina.

- Durante la fase de evaluación no hubo grandes dificultades que afrontar, quizás lo más complicado era conseguir motivar a un número suficiente de usuarios para que participasen en el experimento planteado. En este aspecto, me sorprendió gratamente la respuesta obtenida, ya que muchas personas, tanto de mi entorno más cercano como a través de terceras personas, querían colaborar desde el primer momento que lo propuse, obteniendo además un feedback bastante enriquecedor.

10. TRABAJO FUTURO

Con la realización de este trabajo se plantean una serie de líneas de investigación o trabajos futuros en relación tanto a la solución desarrollada como a las técnicas o herramientas empleadas durante el desarrollo, y que podrían ser planteadas como base para futuros proyectos académicos o trabajos de investigación.

Las diferentes ideas que se plantearán a continuación fueron surgiendo durante las diferentes fases del proyecto, y algunas de las mismas gracias a la aportación directa de los usuarios de evaluación que colaboraron en la última fase de desarrollo del videojuego:

- Creación aleatoria de niveles de juego

La idea es la generación de las pantallas de juego de forma que el mapa resultante tenga sentido en una partida. Un nivel de juego válido debería permitir al jugador poder completar la partida con éxito, de forma que todas las salas y ordenadores sean alcanzables desde la posición de inicio del jugador. Una posible solución a este problema sería la aplicación conjunta de algoritmos propios y técnicas de inteligencia artificial. Otro aspecto bastante interesante sería hacer un estudio de la comparativa entre los niveles generados con estos algoritmos diseñados por el propio investigador y los generados tras la aplicación de técnicas de aprendizaje automático.

- Escalado del juego: enemigos, objetos y minijuegos

Una de las mejoras más comentadas por los usuarios de evaluación era que les habría gustado disponer de mayor variedad de enemigos, objetos y minijuegos, debido a que se les hacía algo repetitivo el propuesto en la solución. Por lo tanto, una posible línea de trabajo sería la ampliación de los elementos comentados. Respecto a los enemigos, un trabajo interesante sería el de incluir otros con diferentes características físicas, variar los sentidos que disponen, los ataques que realizan, o incluso de la forma en la que se mueven por el nivel. Respecto a los objetos, se podrían incluir nuevos tipos de armas y objetos de utilidad, que incluyan mecánicas diferentes a las actuales como, por ejemplo, granadas cegadoras. En cuanto a nuevos minijuegos, la única restricción sería el tiempo medio para su resolución, ya que no debería exceder mucho respecto a los minijuegos actuales de descifrado de claves.

- Aplicación de nuevas técnicas de IA para la creación de agentes inteligentes

Ya sea siguiendo el modelo de arquitectura híbrida planteado en este trabajo, o planteando otro tipo de técnicas de inteligencia artificial, como podrían ser técnicas de aprendizaje automático, se podrían construir otros tipos de enemigos con comportamientos inteligentes diferentes a los actuales. Un aspecto que sería bastante interesante de estudiar para el juego sería la aplicación de sistemas multiagente o agentes colaborativos, de forma en que los diferentes enemigos compartiesen información, distribuyesen roles, o planificasen las acciones de forma conjunta para entorpecer el objetivo del jugador.

- Función multijugador: arquitectura cliente-servidor

Otro aspecto comentado por algunos de los usuarios de evaluación fue que les habría parecido muy interesante poder competir contra otros jugadores de forma simultánea. Esta línea de trabajo probablemente requeriría revisar aspectos de diseño del juego ya que habría que replantear de alguna forma los objetivos a realizar por el jugador durante la partida. Sin embargo, sería bastante interesante ya que implicaría la utilización de otro tipo de infraestructura, como podría ser utilizando una arquitectura de tipo cliente-servidor, con las dificultades técnicas que ello conllevaría: sincronización y comunicación en ambas direcciones

- Adaptación del juego para smartphones

Es una realidad que cada vez tiene un mayor auge la creación de videojuegos para plataformas móviles. Un posible trabajo sería la adaptación del videojuego creado para este tipo de plataformas, para la que quizás habría que hacer un rediseño de la parte estética del juego y sobre todo en los controles del mismo.

11. LISTA DE REFERENCIAS

[1] J. Bella. "Las 8 claves de la filosofía Nintendo", 3DJuegos, [En línea] Disponible en: https://www.3djuegos.com/juegos/articulos/1213/0/las-8-claves-de-la-filosofia-de-nintendo/ (acceso: 11 de julio de 2018).

[2] AEVI, Asociación Española de Videojuegos. *"Anuario 2017. Anuario de la Industria del Videojuego"*. Lugar de publicación: aevi.org.es. [En línea] Disponible en: http://www.aevi.org.es/web/wp-content/uploads/2018/06/AEVI_Anuario2017.pdf (acceso: 11 de julio de 2018).

[3] INE, Instituto Nacional de Estadística. *"Cifras de Población a 1 de enero de 2017. Estadística de Migraciones 2016. Datos Provisionales".* Lugar de publicación: ine.es. [En línea] Disponible en: http://www.ine.es/prensa/cp_2017_p.pdf (acceso: 12 de julio de 2018)

[4] B. Sheffield. "GDC Casual Summit: Blue Fang's Meretzky Defines Casual Games", Gamasutra, The Art & Business of Making Games. [En línea] Disponible en: https://www.gamasutra.com/php-bin/news_index.php?story=17443 (acceso: 12 de julio de 2018)

[5] AEVI, Asociación Española de Videojuegos. *"El sector de los videojuegos en España: impacto económico y escenarios fiscales"*. Lugar de publicación: aevi.org.es. [En línea] Disponible en: http://www.aevi.org.es/web/wp-content/uploads/2018/01/1801_AEVI_EstudioEconomico.pdf (acceso: 15 de julio de 2018)

[6] J. Grill. "The State of Indie Gaming", Gamasutra, The Art & Business of Making Games. [En línea] Disponible en: http://www.gamasutra.com/view/feature/132041/the_state_of_indie_gaming.php?page=1 (acceso: 6 de diciembre de 2018)

[7] R. Cobbett. "Is indie gaming the future?", Techradar, the source for tech buying advice. [En línea] Disponible en: https://www.techradar.com/news/gaming/is-indie-gaming-the-future--716500 (acceso: 6 de diciembre de 2018)

[8] A. Turing. "*Computing Machinery and Intelligence",* p. 433-460. Lugar de publicación: csee.umbc.edu. [En línea] Disponible en: https://www.csee.umbc.edu/courses/471/papers/turing.pdf (acceso: 10 de diciembre de 2018)

[9] "Git Handbook", Github, [En línea] Disponible en: https://guides.github.com/introduction/git-handbook/ (acceso: 26 de junio de 2018)

[10] "Git - Manual de usuario Versión 1", Fiqus, cooperativa de software. [En línea] Disponible en: http://blog.fiqus.webfactional.com/wp-content/uploads/2014/05/git_manual.pdf (acceso: 26 de junio de 2018)

11. LISTA DE REFERENCIAS

[11] “GitLab Enterprise Edition”, GitLab, [En línea] Disponible en: https://gitlab.com/ help (acceso: 27 de junio de 2018)

[12] J. Torrado, M. A. Álvarez. “Introducción a GitLab”, DesarrolloWeb, [En línea] Disponible en: https://desarrolloweb.com/articulos/ introduccion-gitlab.html (acceso: 27 de junio de 2018)

[13] “Balsamiq Mockups - Quick Guide”, TutorialsPoint, [En línea] Disponible en: https://www.tutorialspoint.com/balsamiq_mockups/balsamiq_mockups_quick_guide.ht m (acceso: 20 de julio de 2018)

[14] F. Martinig. “BOUML - Free Unified Modelling Language (UML) & Code Generation Tool”, Methods & Tools, [En línea] Disponible en: http://www.methodsandtools.com/ tools/bouml.php (acceso: 31 de julio)

[15] “Introduction to the Diagrams of UML 2.X”, Agile Modeling, [En línea] Disponible en: http://agilemodeling.com/essays/umlDiagrams.htm (acceso: 31 de julio)

[16] “About YoYo Games”, YoYo Games, [En línea] Disponible en: https://www.yoyogames.com/about (acceso: 30 de junio de 2018)

[17] “GameMaker: Studio - User Manual”, YoYo Games, [En línea] Disponible en: https://docs.yoyogames.com/ source/dadiospice/ (acceso: 1 de julio de 2018)

[18] L. Johnson. “Lone Survivor: Director’s Cut Review”, IGN. [En línea] Disponible en: http://www.ign.com/articles/2013/10/04/lone-survivor-directors-cut-review (acceso: 12 de julio de 2018)

[19] “Lone Survivor: The Director’s Cut”, Metacritic. [En línea] Disponible en: https://www.metacritic.com/game/playstation-vita/lone-survivor-the-directors-cut (acceso: 12 de julio de 2018)

[20] R. McCaffrey. “Mark of the Ninja Review”, IGN. [En línea] Disponible en: https://www.ign.com/articles/2012/09/07/mark-of-the-ninja-review (acceso: 12 de julio de 2018)

[21] “Mark of the Ninja”, Metacritic. [En línea] Disponible en: https:// www.metacritic.com/game/pc/mark-of-the-ninja (acceso: 12 de julio de 2018)

[22] V. Ingenito. “Stealth Inc: A Clone in the Dark Review”, IGN. [En línea] Disponible en: https://www.ign.com/articles/2013/07/27/stealth-inc-a-clone-in-the-dark-review (acceso: 13 de julio de 2018)

[23] “Stealth Inc: A Clone in the Dark”, Metacritic. [En línea] Disponible en: https://www.metacritic.com/game/playstation-vita/stealth-inc-a-clone-in-the-dark (acceso: 13 de julio de 2018)

[24] J. A. Rodríguez. "Dead by Daylight Análisis", IGN. [En línea] Disponible en: https://es.ign.com/dead-by-daylight-pc/105237/review/dead-by-daylight-analisis-para-pc (acceso: 13 de julio de 2018)

[25] "Dead by Daylight", Metacritic. [En línea] Disponible en: https://www.metacritic.com/game/pc/dead-by-daylight (acceso: 13 de julio de 2018)

[26] R. McCaffrey. "INSIDE Review", IGN. [En línea] Disponible en: https://www.ign.com/articles/2016/06/28/inside-review (acceso: 14 de julio de 2018)

[27] "INSIDE", Metacritic. [En línea] Disponible en: https://www.metacritic.com/game/pc/inside (acceso: 14 de julio de 2018)

[28] C. Harris. "Professor Layton and the Curious Village Review", IGN. [En línea] Disponible en: https://www.ign.com/articles/2008/02/13/professor-layton-and-the-curious-village-review (acceso: 15 de julio de 2018)

[29] "Professor Layton and the Curious Village", Metacritic. [En línea] Disponible en: https://www.metacritic.com/game/ds/professor-layton-and-the-curious-village (acceso: 15 de julio de 2018)

[30] M. Rouse. "Intelligent Agent (AI)", Techtarget. [En línea] Disponible en: https://whatis.techtarget.com/definition/intelligent-AI-agent (acceso: 28 de diciembre de 2018)

[31] "About Freesound", Freesound.org. [En línea] Disponible en: https://freesound.org/help/about/ (acceso: 15 de octubre de 2018)

[32] M.R. Corripio Gil Delgado. "La protección de las medidas tecnológicas y de la información para la gestión de los derechos.", Revista Icade, Revista de las Facultades de Derecho y Ciencias Económicas y Empresariales, n. 78, p. 193-206, jul. 2012. ISSN 2341-0841. [En línea] Disponible en: http://revistas.upcomillas.es/index.php/revistaicade/article/view/238/178 (acceso: 1 ene. 2019)

[33] P.A. De Miguel Asensio. "*Derecho privado por Internet".* Ed. Thomson Reuters/ Civitas 2015.

[34] F.J. Biurrun Abad. "*Pokémon Go y su impacto en la 'legalidad aumentada'*", Actualidad Jurídica Aranzadi núm.922/2016 parte Nuevas Tecnologías. Ed. Aranzadi 2016.

11. LISTA DE REFERENCIAS

SUMMARY OF THE PROJECT

1. Introduction

1.1. Motivation

The human being, from its first stages of life, has the need to have fun. It has always been said that games or, in general, the activities that produce in the individual some kind of feeling of fun are especially useful for childhood, since they promote the development of physical aptitudes, creativity, emotional intelligence, social skills, in addition to strengthening the personality and the introduction of behavioral values. Games are very useful activities to know the environment that surrounds us and introduce concepts as important as empathy, generosity or cooperation between people.

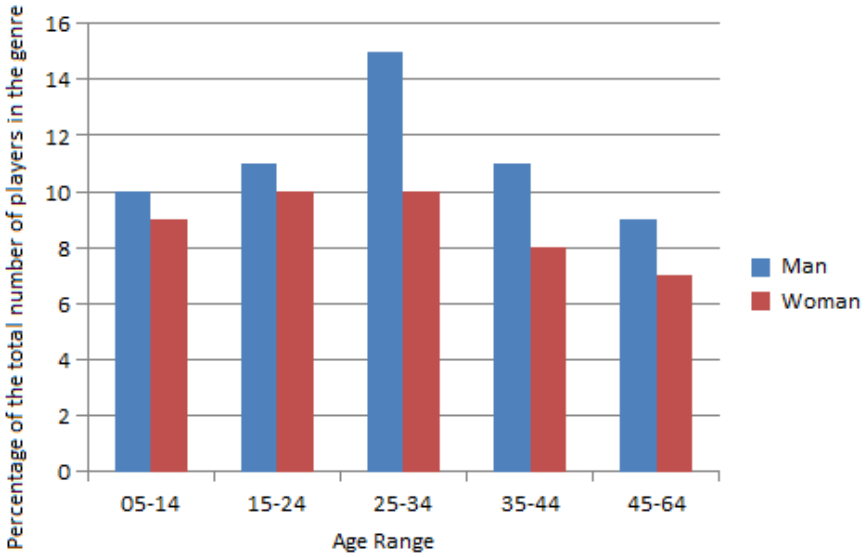
However, there is a tendency to think that as people reach the adult stage, fun or entertainment is unnecessary and not so important for the emotional balance and health of the human being. This really should not be the case, since precisely healthy leisure is one of the best activities to free the mind from everyday stress, from the routine of daily responsibilities, and in short, maintain a balance in the lifestyle that many times it's imposed in today's society.

A great example of fun philosophy, without discriminating any age, is the one applied by the Japanese video game company Nintendo, which can resume that its philosophy is to create products to generate smiles in the players and for which one of their maximum personalities, Shigeru Miyamoto, clearly defined with the phrase "I believe that within each adult resides the heart of a child. The only thing that we have gradually convinced ourselves it that we have to act more like adults".

As a result of the above and leaving aside more abstract aspects, it's a reality that the videogame industry is booming. If we look at the Spanish level, the videogame industry billed in 2017, according to reports written by the Spanish Association of Videogames (AEVI), 1,359 million euros, seeing this figure increased compared to the previous year in almost 17%. Comparing these figures with those obtained in that same year by other sectors of culture and entertainment, such as the film industry, which billed 232 million euros, we can highlight even more the relevance of this sector and that in many occasions it is not appreciated in our society.

This position of relevance that the videogame industry has in society is due, to a greater extent, to a large number of existing players and to the remarkable variety of profiles of players who consume video games on a regular or casual basis. In the same report referred to above, a very eloquent fact is offered regarding this fact, and it's stated that 44% of Spaniards whose age is between 6 and 66 years old play video games, even in a casual way. Taking into account the population data published by the Spanish National Institute of Statistics for the year 2017, it would be concluded that approximately 15,5 million Spaniards play video games.

In the AEVI report itself, data on the distribution of the player population are given, quantifying them by gender and age ranges, as it's shown in the following:



Analyzing the data, we have that most players with respect to the total of players and including both genders are in the range of between 25 and 34 years of age. It can be concluded, moreover, that there is a fairly clear prejudice regarding the predominant age range of the video game player profile. It's usually said that video game players are mostly teenagers and young people without a job and family responsibility.

In part, the graph gives the reason for this, to the extent that a large part of the players corresponding to these age ranges. However, there is another large group of players with ages over 25 and in all the analyzed age ranges. Thanks to this information, it can be concluded that there is a great variety of player profiles and potential clients within the video game industry.

It's true that the current video game market has a very wide offer to cover all types of player profiles. However, it's important to highlight that player profiles who have less time to devote entertainment, whether for work and/or family reasons, and that aren't less relevant when it comes to proposing and designing new games.

A very interesting concept regarding a specific type of player, and for those who have developed a good part of the video game industry, is what is known as a casual player, which was already spoken in 2008 at the Conference of Game Developers (GDC). In this conference, the organizers defined the concept of the casual game as "that game aimed at people for whom video games aren't an important area of interest, or at least, of much less important than for a regular player". In the same way, it has arrived at a series of main characteristics that usually share the casual games, as they are the following:

- Low entry barriers: low loading time, the absence of complex installations and long tutorials.

- Indulgent: give certain clues and not penalize excessively the mistakes made by the player during the course of a game, at least at the lowest levels of difficulty or in the first levels of the game.

- Short game sessions: a game session shouldn't require more than 10-15 minutes.

- Highly redeemable: if the game is of short duration it will have to be replayable to justify its purchase.

- The depth of the game revealed gradually: if the game is introducing new elements that are gradually and explained to facilitate their understanding to the player.

The problem that is intended to be solved in this Final Degree Project is the design and implementation of a videogame that offers a market option for that profile of casual players and directed, above all, to players within an age range starting from the 25 years on, that don't have a great demand for both the videogame graphics but for the most playable and fun aspect.

1.2. Objectives

With the completion of this Final Degree Project, which intention is the design and development of a videogame with the characteristics discussed in the previous subchapter, the following objectives are sought:

1- Expand the offer of casual games existing in the market

The videogame to develop with this work will be directed mainly to a players profile who still enjoy video games, but for work or family reasons don't have so much time to devote to this leisure option. A business opportunity has been detected within the casual videogame market that is intended to be exploited.

2- Create a game of easy portability

It will be sought that, in a simply way, the videogame can be exported to other platforms that also have a lot of interest for the casual player's profile, such as smartphones or portable game consoles. To achieve this goal, the chosen programming tool will be very important, which for the present work will be Game Maker: Studio 2. This tool, among other features, allows you to deploy the game created in different platforms depending on the license purchased.

3- Create an easily scalable game

It will be sought that the design of the game and its subsequent implementation allow a simple scaling, both in scenarios, types of enemies, objects, puzzles and other elements of the game.

4- Application of a traditional software development methodology

It’s interesting to see how the typical phases of analysis, design, implementation, and evaluation, can be applied to a product less common in the academic training of a computer engineer, and that can serve as a reference for similar future projects.

2. Solution

2.1. Solution definition

Once studied the problem and reach the conclusion of the potential of casual players existing in the national panorama of Spain, it is proposed as a solution the design and implementation of a video game aimed at that target audience. The goal is to build a simple and intuitive game, while being fun, and with game sessions of no more than 12 minutes per game.

With these premises, we will briefly define the main features and mechanics of the proposed game

10. It’s a game that combines the genres of action and puzzles.

11. The objective of the player is to get away from a complex, which is completely closed and with a certain number of enemies. You must do it within a time limit before the entire complex is destroyed. To be able to escape from the complex you will have to find and solve certain puzzles while avoiding being defeated by the enemies that will be found by the complex. It will be the player's decision if he chooses to try to sneak between the enemies or try to buy time by facing them.

12. In case that the player needs or wishes to confront the enemies, a series of objects (weapons, ammunition, and other useful objects) scattered throughout the complex are available and will help in the resolution of the game.

13. The puzzles to be solved during the course of the game will be presented in a simple and intuitive way so that the resolution of the player does not take more than 2 minutes per puzzle on average. It will consist of deciphering a four-digit numerical code in which each figure can go from values from "1" to "6" by having unlimited attempts. After confirming the player a resolution attempt, clues will be offered about how many figures are well placed, and if any figure is part of the correct code but is not in the right place. These clues will be given visually by a quite intuitive symbol code. When the numerical code is correctly entered, the puzzle will be considered resolved and the player will be alerted.

14. Enemies will behave intelligently based on the information obtained through the sense of sight and hearing. They will go from a normal state of movement to a state of hostility if they have the player in visual range or if they hear a noise, usually produced by firearms, at a certain distance.

15. The game will seek the fun of the player by combining the feeling of tension that will generate survive within the complex full of enemies with fairly limited equipment, along with the resolution of the puzzles using the logic ability. All in

it in real time and always keeping in mind that it must be resolved before the time counter reaches zero.

16. The game will have several game modes escalating the difficulty. Although the main mechanics of the game doesn`t change, it will change the number of enemies that will appear on the stage, the number of puzzles to solve, and the scenario itself. In any case, the maximum starting resolution time will not change.

17. Following one of the current trends in video games, such as the resurgence of games with classic or retro graphics, also known as Pixel Art games, the game is presented in this line. In this type of games, the search for realism in the visual section is in the background, giving a higher priority to the gameplay.

18. It will work on a PC platform, with a Windows operating system.

An important part of the definition of the solution is to take into account that the different services offered by the Game Maker tool will be used: Studio 2.

2.2. Requirements specification

Generally, the main source of the requirements in most of the software developed today is part of the client or the end user of the application. However, for the present work and due to the nature of the software developed, in the case of the creation and design of a videogame itself, it will be considered that the main source of the requirements is the author of the work. This performs the work of an analyst, designer, and programmer of the generated software.

The list of requirements used in this work divides them into two categories: capacity requirements and restriction requirements. The capacity requirements serve to define what you want to create, or otherwise, what features the video game will have. The restriction requirements, on the other hand, serve to define how the video game will be constructed, establishing a series of limitations on the capacity requirements. Keep in mind that the level of detail in the description of requirements that are made in this list should not be excessive since we are in the analysis phase. This level of detail will be deepened in the design phase.

Below are two examples of requirements, one of capacity and one of restriction, extracted from the list of requirements generated as a result of the analysis phase. In total, there are defined 24 capacity requirements and 9 restriction requirements.

RC-01	Main screen		
Description:	The game will have a main screen from which the player can start a new game, see the best scores and check the credits.		
Priority:	☑ High ☐ Medium ☐ Low	Source:	Analyst
Need:	☑ Essential ☐ Convenient ☐ Optional		
Verifiability:	☑ High ☐ Medium ☐ Low		

RR-01	Platform		
Description:	The game will work exclusively for the PC platform.		
Priority:	☑ High ☐ Medium ☐ Low	Source:	Analyst
Need:	☑ Essential ☐ Convenient ☐ Optional		
Verifiability:	☑ High ☐ Medium ☐ Low		

2.3. Solution design

The objective of this chapter of game design is to detail some aspects related to the context of the game, which was not described in detail in the analysis phase, such as the synopsis, aspects of gameplay and the mentality that is sought in the player when playing a game to the designed game.

In the game designed, the aim is to provoke a mentality in the player that is a mixture of several feelings:

- Tension and nerves, produced by being a game where you must complete a mission before a time limit and where there are a number of enemies that we will have to sneak off or face in extreme cases.

- Intelligence and planning, produced in the part of the logical resolution of the decryption puzzles of the keys and the aspects of stealth and exploration of the laboratories trying not to be detected by the enemies.

Another fundamental part of the game design is the construction of the prototypes of the different screens that compose it. Below is an example of defining a prototype screen with the description of the elements that are in it, and that should be taken into account in the implementation phase.

In a game screen, like the one shown above, you can distinguish the following elements:

- The life bar, located in the upper left part of the screen, which will serve to indicate to the player the total remaining life in a fairly intuitive way.
- An icon of the equipped object, located in the second part of the upper part of the screen, which will indicate which object the player is currently equipped with.
- The ammunition or remaining uses of the equipped object.
- An indicator of the number of keys to be deciphered complete and total of the level.
- A remaining time counter expressed in seconds before the game is finished with defeat.
- The level of the game, where the total or a portion of the map is displayed if it occupies more than the screen size of the device. In the previous image, you can see different rooms, some open and another closed. In the case of the closed room, it is not possible to observe the inside of the room until it is opened, as is logical, for what is presented with a default fill color. In the example, you can also see how the enemies and the main character could be, as well as some elements to provide concealment to the character regarding the visual range of the enemies. In some of the rooms, you can find the computers, such as those that can be seen in the image, to decipher the keys and thus open the exit door of the laboratories, which also appears in the image shown in blue and larger than normal doors.

2.4. Solution implementation

In this phase, it is explained how each of the main aspects that make up the created videogame has been implemented from the technical point of view. For each section, we will provide a series of captures taken from the Game Maker: Studio 2 tool, interspersed with screenshots of the game in its definitive version where it is deemed appropriate to help its understanding.

Due to the summary nature of this document, we will only comment on how the implementation of one of the most interesting aspects of the work has been carried out. In particular, we refer to the programming of the artificial intelligence of the enemies.

Of all the typical paradigms that are used for the programming of intelligent agents, we have chosen to use the hybrid paradigm, with the following characteristics:

- The default behavior of an enemy agent is to patrol a certain area of the game level, following a pre-established path.

- This behavior is altered depending on the information extracted by two senses that the enemy agent has: sight and hearing.

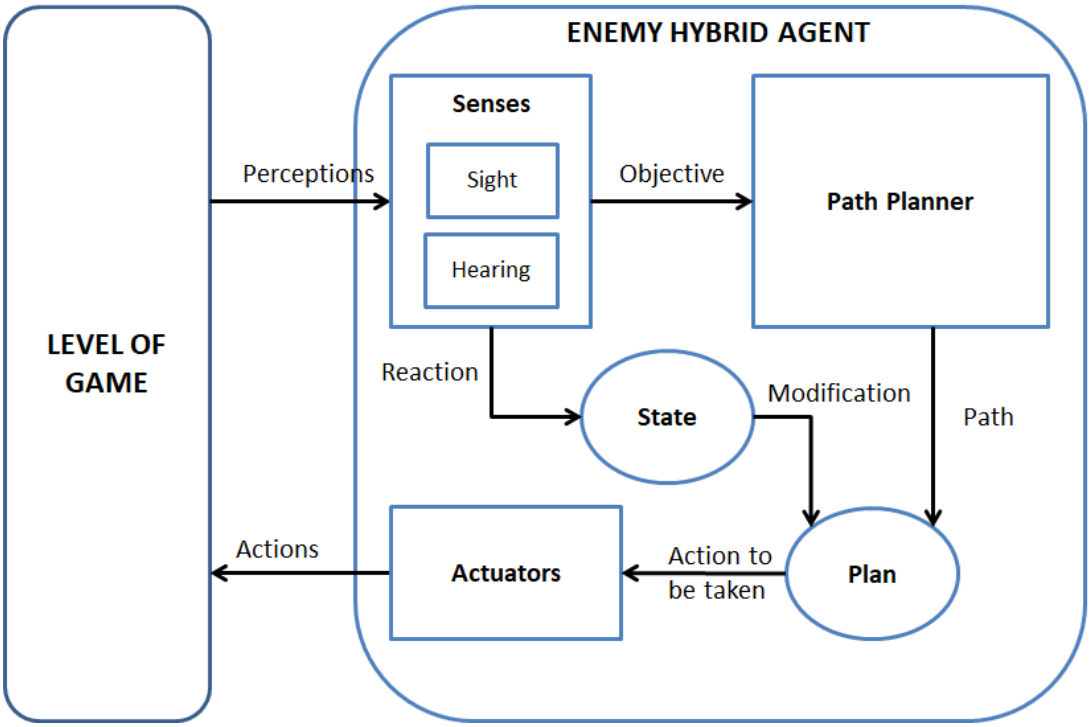
- The sense of sight provides the agent with information about the distance between that agent and the main character. In the case that this distance is less than a value of 300 pixels and there is no opaque object between the two characters, it is considered that the enemy has the protagonist in view, for which he changes his behavior to a state of persecution.

- The sense of hearing provides the agent with information on noise generation by the main character. When the protagonist performs some action that produces noise, such as firing a firearm, detonating a smoke bomb, or running, noise is generated. If the enemy agent is within a distance of fewer than 150 pixels, it is considered that the enemy has an auditory reach to the protagonist, for which he changes his behavior to a state of persecution.

- Once the reactive part of the enemy agent activates the state of persecution, the deliberative part of the agent comes into play: the planning of the path to be traveled between the agent and the protagonist or the source of the noise. The implementation of this part is made from functions provided by the Game Maker tool itself for the search of shorter paths between two objects that are in a game level.

- To control the behavior of the enemies and the different transitions, a state is included, which for the enemy character can take the values of "patrol", "return", "observe", "pursue", "attack", "paralyzed" and "hurt."

Next, a figure that shows the scheme of the hybrid architecture finally implemented in an enemy agent is presented, being able to observe how the internal and external elements interact with the agent itself:



2.5. Solution evaluation

For the evaluation phase of the game, the idea has been to choose users that are included in the target audience of the game, since they can provide much more reliable information, to take into account to evaluate the product, and provide constructive criticism about it. Even so, some inexperienced users have been taken into account in video games or computer use since it has been considered that they could contribute opinions from a different point of view.

In total, 25 users participated in the evaluation phase, which was mainly obtained through three channels: family, friends, and contacts through social networks, mainly through a personal account of the social network Twitter. All of them have been informed of the academic purpose of the experiments, of the absence of lucrative aims of said task and of the anonymous treatment of the data obtained through the questionnaires and the interviews. From the point of view of the evaluation, the only requirement that has been asked has been to respond honestly and as critically as possible within their knowledge or experience with video games.

The experiments carried out have been of two types: supervised and unsupervised. In the supervised experiment, the creator himself has been observing, recording results and resolving doubts with the user present. In the unsupervised experiment, on the contrary, it consists of a test of the game without the participation of the expert. To do this, it has been provided to all users who would like to participate a small guide to avoid any kind of doubt and thus achieve that the total of the experiments was carried out in the most homogeneous way possible. It is important to say that most of the experiments have

been done in an unsupervised way, due to the availability of time and for greater convenience for the users.

Once the experiment has been carried out on each user, the questionnaire is filled out by the user, and optionally, a small interview. Both techniques have as objective to obtain from the user the game experience and the opinion on different aspects of the product, such as the gameplay, the graphics aspect, the difficulty in the game controls, to extract possible improvements or aspects that have been missing.

The analysis of the results extracted from the different evaluation techniques has been done from two points of view. On the one hand, a quantitative analysis of the responses of the questionnaires, and on the other, a qualitative analysis of the answers obtained from the interviews.

3. Conclusions

Once this final degree project has been developed, it's of particular interest to write conclusions about the objectives set in the first phases of the work and how they are reflected in the results finally obtained with the development of the proposed solution.

For greater clarity we will choose to list each of those objectives set out at the beginning of the work and justify its fulfillment.

1- Expand the offer of casual games existing in the market

The game has been implemented satisfactorily fulfilling all the requirements raised in the analysis phase. As for the casual aspect of the game, it's perfectly reflected in several of the requirements of the videogame implemented. For example, the duration of a game doesn't exceed 12 minutes, the game has several levels of difficulty, it does not have an excessively high learning curve, it has quite intuitive interfaces, and the graphics of the game are in line with the habitual trends in the most successful casual games nowadays, such as retro graphics or pixel art.

2- Create a game of easy portability

While it's true that the license acquired only allowed to deploy the game on the PC platform, during the design and implementation of the game, this characteristic was taken into account, not overcomplicating aspects to be taken into account in portability, such as the controls of the game, so that it's easy to change the events that control the character or the different menus of the game screens.

3- Create an easily scalable game

When using an object-oriented programming paradigm, it's quite intuitive to observe how the game elements discussed above have been implemented and use them as templates for the creation of other custom elements. Therefore, the resulting code is easy to read and understand for future improvements of the game, in aspects such as the expansion of game levels, types of enemies and objects.

4- Application of a traditional software development methodology

A cascade development methodology with feedback has been used, fulfilling each of the development phases according to the typical tasks of any other type of software. The main differences between the different typical activities in the development methodologies for a traditional software and a video game such as the one developed have been located in the design phase, where perhaps more prominence has been given to activities such as the design of prototypes and the definition of game mechanics, typical of this type of software.

Once each of the proposed objectives has been analyzed, it can be concluded that the work performed complies with them and successfully solves the problem posed at the beginning of the work.

Regarding the main difficulties encountered during the execution of this work, the following can be mentioned:

- During the pre-analysis phase of the solution, based on pure creativity, a series of iterations had to be carried out until a solution was found that could be considered within a reasonable time frame within the context of a Final Degree Project and that, at the same time, it could give a successful response to the objectives set at the beginning of the work.

- During the transition from the analysis phase to the design phase, some adjustments had to be made to the requirements raised at first, because they weren't correctly reflected in the next phase of development.

- In the solution implementation, one of the main difficulties encountered was the implementation of the game graphics, mainly due to my own inexperience with the use of graphics tools at a more advanced level. Another of the most complex elements to implement, but at the same time, a very rewarding challenge to overcome was the creation of artificial intelligence of the characters controlled by the machine.

- During the evaluation phase there were no great difficulties to face, perhaps the most complicated was to motivate enough number of users to participate in the proposed experiment. In this regard, I was pleasantly surprised by the response obtained, like many people, both from my immediate environment and through third parties, wanted to collaborate from the first moment I proposed it, also obtaining enriching feedback.

4. Future works

With the realization of this work, a series of research lines or future works are proposed in relation both to the developed solution and to the techniques or tools used during the development, which could be proposed as a basis for future academic projects or research projects.

The different ideas that will arise next were arising during the different phases of the project, and some of them thanks to the direct contribution of the evaluation users who collaborated in the last phase of videogame development:

- Random creation of game levels

The idea is to generate the game screens so that the resulting map makes sense in a game. A valid level of play should allow the player to complete the game successfully so that all rooms and computers are reachable from the player's starting position. A possible solution to this problem would be the joint application of proprietary algorithms and artificial intelligence techniques. Another interesting aspect would be to make a study of the comparison between the levels generated with these algorithms designed by the researcher himself and those generated after the application of automatic learning techniques.

- Scaling the game: enemies, objects, and mini-games

One of the improvements most commented by the evaluation users was that they would have liked to have a greater variety of enemies, objects, and mini-games because they were doing something repetitive proposed in the solution. Therefore, a possible line of work would be the expansion of the elements discussed. Regarding the enemies, an interesting job would be to include others with different physical characteristics, vary the senses they have, the attacks they perform, or even the way they move through the level. Regarding the objects, new types of weapons and useful objects could be included, which include different mechanics than the current ones, such as, for example, blinding grenades. As for new mini-games, the only restriction would be the average time for its resolution, since it should not exceed a lot with respect to the current key decryption minigames.

- Application of new AI techniques for the creation of intelligent agents

Whether following the hybrid architecture model proposed in this paper, or proposing another type of artificial intelligence techniques, such as automatic learning techniques, other types of enemies could be built with intelligent behaviors different from the current ones. One aspect that would be quite interesting to study for the game would be the application of multi-agent systems or collaborative agents, in a way in which the different enemies shared information, distributed roles, or planned the actions together to hinder the player's objective.

- Multiplayer function: client-server architecture

Another aspect commented by some of the evaluation users was that they would have found it very interesting to be able to compete against other players simultaneously. This line of work would probably require revising aspects of the game design since the objectives to be performed by the player during the game would have to be reconsidered in some way. However, it would be quite interesting since it would involve the use of another type of infrastructure, such as using a client-server architecture, with the technical difficulties that this would entail: synchronization and communication in both directions.

- Adaptation of the game for smartphones

It's a reality that increasingly has the creation of video games for mobile platforms. A possible work would be the adaptation of the videogame created for this type of platforms, for which perhaps it would be necessary to make a redesign of the aesthetic part of the game and especially in its controls.